**Punctuality Drive Using Face Recognition**

*A project report submitted*

*to*

**Dr. A.P.J. ABDUL KALAM TECHNICAL**

**UNIVERSITY LUCKNOW**

*For partial Fulfillment of the Requirement for the*

*Award of the Degree*

*of*

**MASTER OF COMPUTER APPLICATION**

*by*

**SADIQ ALI- 2000270140045**

**SHIVAM SHUKLA - 2000270140048**

**SURYANSH SHUKLA - 2000270140056**

**RAHUL YADAV - 2000270140042**

**TUSHAR SHARMA- 2000270140059**

**Under Guidance of**

**Mrs- Indu Verma**

****

**AJAY KUMAR GARG ENGINEERING**

**COLLEGE-MCA**

**GHAZIABAD**

**2021-22**

**COLLEGE CERTIFICATE**

****

This is to certify that **Sadiq Ali (2000270140045) , Shivam Shukla (2000270140048) , Suryansh Shukla (2000270140056) , Rahul Yadav (2000270140042) and Tushar Sharma (2000270140059)** are thebonafide students of **Ajay Kumar Garg Engineering College-MCA, Ghaziabad,** and have submitted this project report on “**Punctuality Drive Using Face Recognition**”for fulfillment of the requirements of **Dr. A.P.J**. **Abdul Kalam Technical University, Lucknow** for awarding the degree of **“MASTER OF COMPUTER APPLICATION”.**

Dr. B. K. Sharma

Principal,

Ajay Kumar Garg Engineering College-MCA, Ghaziabad

**ACKNOWLEDGEMENT**

We would like to express your sincerest gratitude to all the people who have contributed towards the successful completion of our project. We would like to extend our heartfelt thanks to the Head of Master of Computer Application Department Dr B.K. Sharma, for nurturing a congenital yet competitive environment in the department, which motivates all the users to pursue higher goals. We want to express our special gratitude to our guide “Prof Indu Verma” , Department of MCA, Ajay Kumar Garg Engineering College, Ghaziabad for her constant support, guidance, encouragement and much needed motivation. Her sincerity, thoroughness and perseverance has been a constant source of inspiration for us. Last but not the least, we would like to extend our thanks to all the teaching and non-teaching staff members of our department and to our colleagues who helped us in the completion of the project.

SADIQ ALI- 2000270140045

SHIVAM SHUKLA - 2000270140048

SURYANSH SHUKLA - 2000270140056

RAHUL YADAV - 2000270140042

TUSHAR SHARMA- 2000270140059

**LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE**

**DFD** : Data Flow Diagram

**LBPH**: Local Binary Pattern Histogram.

**OpenCV**: Open-Source Computer Vision Library.

**Tkinter**: Tkinter is a Python binding to the Tk GUI toolkit.

**HaarCascade**: It is an Object Detection Algorithm used to identify faces in an image or a real-time video

**Face Recognition**: Used to detect faces on blur images.

**SQL** : Structured Query Language

|  |  |  |
| --- | --- | --- |
| **LIST OF FIGURES** |  | |
| **Figure Number & Name** | |  | |
|  |  | |
| System Flowchart |  | |
| E-R Diagram |  | |
| Symbols of Data Flow Diagram |  | |
| Level 0 DFD |  | |
| Level 1 DFD |  | |
| Modular Design |  | |
| Context Diagram |  | |

**ABSTRACT**

Face Recognition is a computer application that is capable of detecting, tracking, identifying or verifying human faces from an image or video captured using a digital camera. Although lot of progress has been made in domain of face detection and recognition for security, identification and attendance purpose, but still there are issues hindering the progress to reach or surpass human level accuracy. These issues are variations in human facial appearance such as; varying lighting condition, noise in face images, scale, pose etc. This research paper presents a new method using Local Binary Pattern (LBP) algorithm combined with advanced image processing techniques such as Contrast Adjustment, Bilateral Filter, Histogram Equalization and Image Blending to address some of the issues hampering face recognition accuracy so as to improve the LBP codes, thus improve the accuracy of the overall face recognition system. Our experiment results show that our method is very accurate, reliable and robust for face recognition system that can be practically implemented in real-life environment as an automatic attendance management system.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Content** |  | **Page No.** |
| 1. INTRODUCTION | |  |
| 1.1 | Project Introduction |
| 1.2 | Problem Description |



2. SYSTEM STUDY

2.1 Existing System with limitations

2.2 Proposed System with objectives

2.3 Feasibility Studies

2.3.1 Economical

2.3.2 Technical

2.3.3 Duration

3. SYSTEM ANALYSIS

3.1 System Flowcharts

3.2 E-R Diagrams

3.3 Data Flow Diagram (DFDs)

3.4 Requirement Specifications

4. SYSTEM DESIGN

4.1 Modular Design

4.2 Context Diagram

4.3 Structure Chart

4.4 File / Database Design

4.5 Normalization

4.6 Input / Output form design

4.7 Screen Design

4.8 Report Design

1. SYSTEM IMPLEMENTATION
2. 5.1 H/W and S/W Requirement

5.2 User Interface Design

5.3 Coding

5.3.1 List of Programs

5.3.2 Inputs

53.3. Outputs

6. SYSTEM TESTING

6.1 Preparation of Test Data & Test Oracle

6.2 Unit Testing, Class Testing

6.3 Testing with dummy / live data

7. SYSTEM SECURITY

7.1 Checks and Control

7.2 Encryption, Secure

1. LIMITATION AND SPECIAL FEATURE
2. CONCLUSION
3. FUTURE SCOPE
4. BIBLIOGRAPHY

**CHAPTER - 1**

**INTRODUCTION**

* 1. **Project Introduction**

This is punctuality drive software where we check which student is punctual to the college and not punctual to the college. We recognize the face of the student and then mark the attendance and give rewards to the students.

Punctuality Drive is a way of identifying or confirming an individual's identity using their face. Facial recognition systems can be used to identify people in photos, videos, or in real-time. Facial recognition is a category of biometric security

* 1. **Problem Description**

The main problem was to solve the process of punctual students and hence give them some rewards for being punctual to the college.

In this project we take sample of students face and store the data in database.Then we collect the students details so that when he enters the college his attendence can be marked. Then we train the data of the students.

We can delete and update any students record..

As soon as the students enters the college his or her attendence gets marked. If the student is on time or the student is late , his attendence time will be monitored and a report will be made.

If the student is consistently coming to college on time then his report will be monitored and hence award will be given to the students who is punctual

**CHAPTER - 2**

**SYSTEM STUDY**

**2.1 Existing System with limitations:-**

* Requires high bandwidth.
* Have internet connectivity.
* Should have Proper device

As with any technology, there are potential drawbacks to using facial recognition, such as threats to privacy, violations of rights and personal freedoms, potential data theft and other crimes. There's also the risk of errors due to flaws in the technology.

Detection is vulnerable. While face detection provides more accurate results than manual identification processes, it can also be more easily thrown off by changes in appearance or camera angles. A potential breach of privacy.

**2.2 Proposed System with objectives :-**

The objective of PUNCTUALITY DRIVE is to reward the students who are punctual throughout the semester , hence make a report of all the late comers and punctual students.

**Face Detection**

The camera detects and locates an image of a face, either alone or in a crowd. The image may depict the person looking straight at the camera or in profile

**Mark Students Punctuality and defaulter**

An automated attendance system is used for marking human face recognition in real time for college to mark the attendance for their Punctuality and defaulter students. Smart automated attendance marking using real time face recognition is a real world solution which comes with day to day activities of handling Students

**2.3 Feasibility Studies**

After doing the project Punctuality Drive Using Face Recognition System, study and analyzing all the existing or required functionalities of the system, the next task is to do the feasibility study for the project. All projects are feasible - given unlimited resources and infinite time.

Feasibility study includes consideration of all the possible ways to provide a solution to the given problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that

future changes can be easily done based on the future upcoming requirements

**2.3.1 Economical**

This is a very important aspect to be considered while developing a project. We decided on the technology based on the minimum possible cost factor.

All hardware and software cost has to be borne by the organization. Overall we have estimated that the benefits the organization is going to receive from the proposed system will surely overcome the initial costs and the later on running cost for the system.

**2.3.2 Technical**

This included the study of function, performance and constraints that may affect the ability to achieve an acceptable system. For this feasibility study, we studied complete functionality to be provided in the system, as described in the System Requirement Specification (SRS), and checked if everything was possible using different types of frontend and backend platforms.

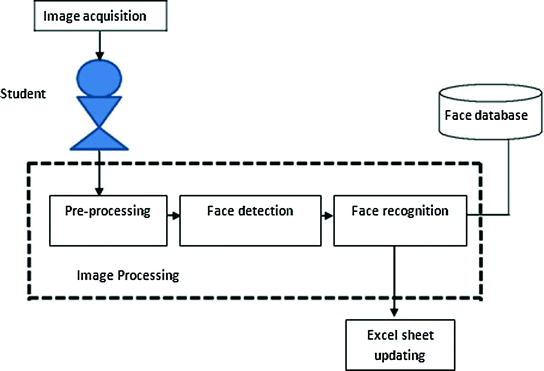
**2.3.3 Duration**

No doubt the proposed system is fully GUI based that is very user friendly and all inputs to be taken all self-explanatory even to a layman. Besides, a proper training has been conducted to let know the essence of the system to the users so that they feel comfortable with new system. As far our study is concerned the clients are comfortable and happy as the system has cut down their loads and doing

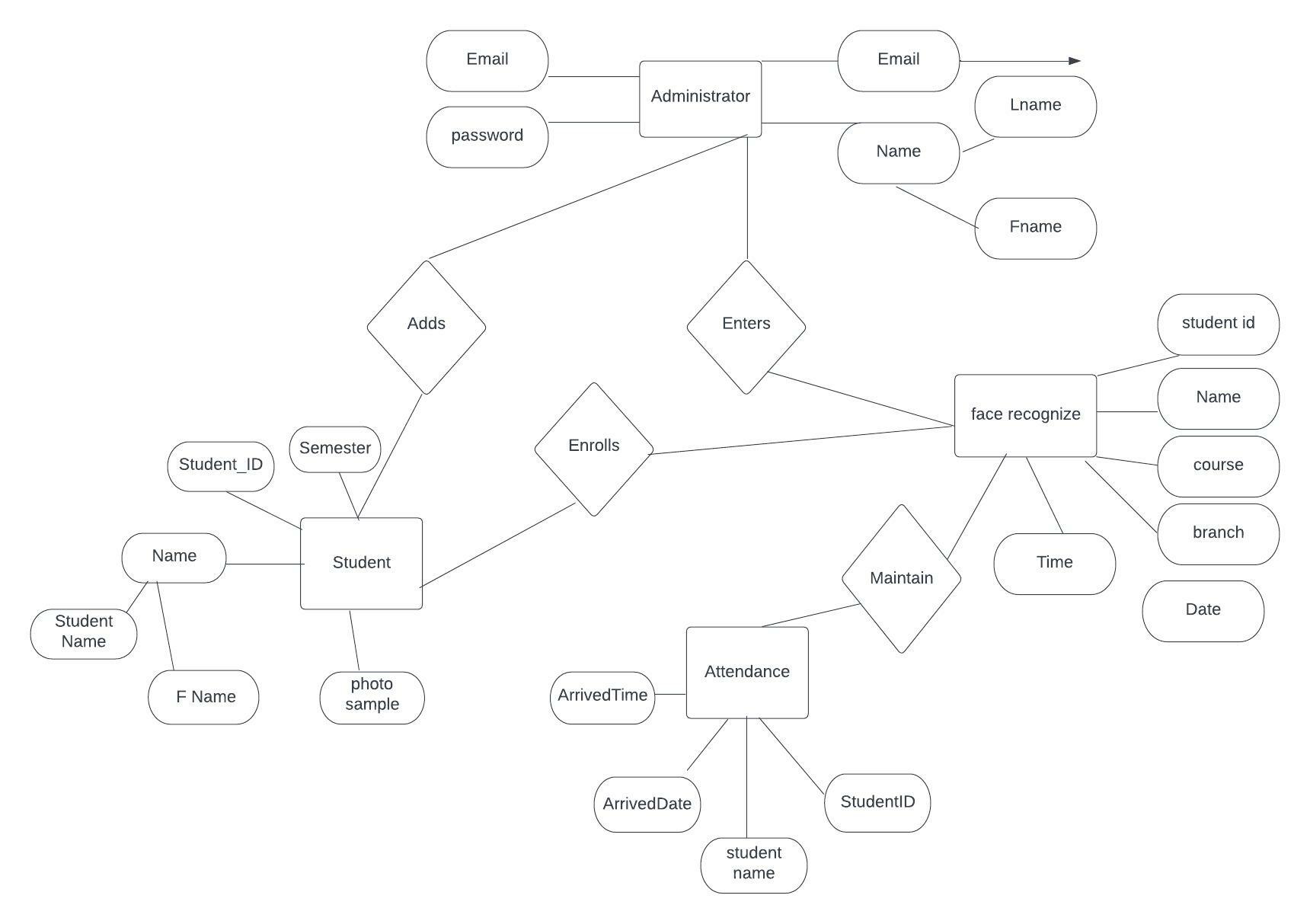
**CHAPTER - 3**

**SYSTEM ANALYSIS**

1. SYSTEM ANALYSIS
   1. **System Flowcharts**



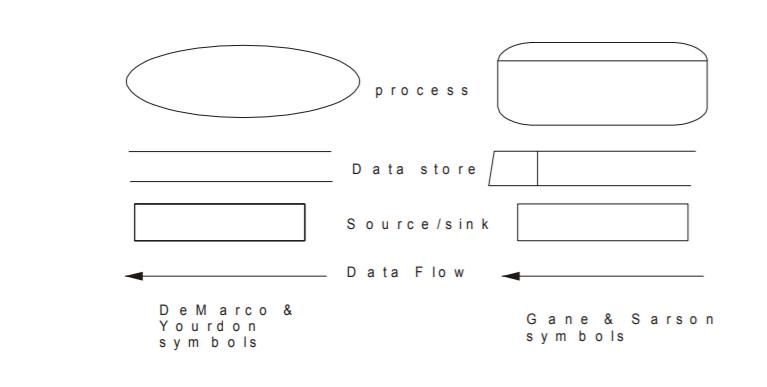
**3.2 E-R Diagrams :-**

****

**3.3 DATA FLOW DIAGRAMS**

Data flow diagram is the starting point of the design phase that functionally decomposes the requirements specification. A DFD consists of a series of bubbles joined by lines. The bubbles represent data transformation and the lines represent data flows in the system. A DFD describes what data flows rather than how they are processed, so it does not use hardware, software and data structure. A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design). A data flow diagram (DFD) is a significant modeling technique for analyzing and constructing information processes. DFD literally means an illustration that explains the course or movement of information in a process. DFD illustrates this flow of information in a process based on the inputs and outputs. A DFD can be referred to as a Process Model.

Unlike detailed flow charts, DFDs don’t supply detailed descriptions of modules that graphically describe a system’s data and how the data interact with the system. Data flow diagram number of symbols and the following symbols are by DeMarco.



**Fig 3.3.1 Symbols of Data Flow Diagrams**

**Zero Level DFD**

Manage Student Details

Admin Login

Report Generation System

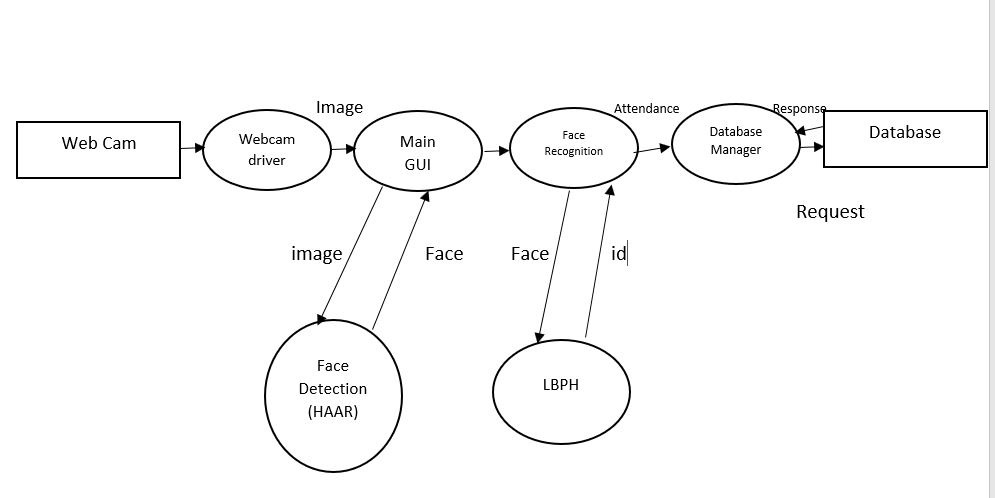
Add Student Photos

Face Recognition System

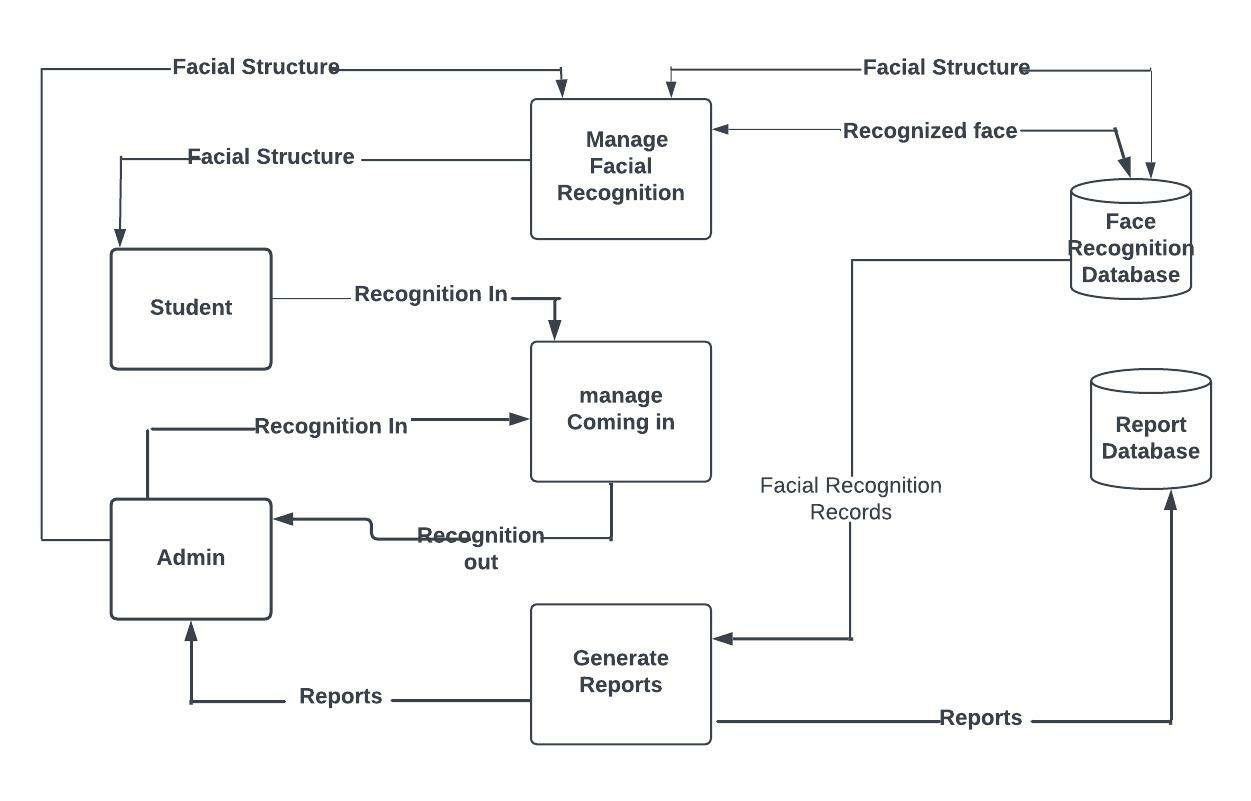
Mark Attendance

Train face

Level 1 DFD



**DFD LEVEL 2**

****

**3.4 Requirement Specifications** :- The following list presents the external interface requirements:-

The product requires good graphics usage.

The product does not require usage of sound or animation.

The hardware and operating system requires a screen resolution not more than 320 x 240 pixels.

Sound is not an essential feature but it can be considered for future variants of the system where in the user will be greeted when the attendance is marked and alerted when not marked.

**Software Requirements:**

**Hardware Interface:**

## Technology/Platform/Tools used

**Technology:**

* Django
* OpenCV
* Dlib
* Open-Source Face Recognition Library
* SQLITE Database.
* Python

**Platform:**

* Windows
* Linux

**Tools:**

* Visual Studio Code / PyCharm

**Non-Functional Requirements-**

A non-functional requirement is a specification that describes the system’s operation capabilities and constraints that enhance its functionality. These may be speed, security, reliability, etc. We’ve already covered different types of [software requirements,](https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/) but this time we’ll focus on non-functional ones, and how to approach and document them.

**Performance Requirements-**

The product shall be based on the web and has to be run from a web server. The product shall take initial load time depending on internet connection strength which also depends on the media from which the product is run. The performance shall depend upon hardware components of the client/customer/end user.

**Security Requirements-**

After entering the password and user id the user can access his profile. The details of the user must be safe and secure.

**Compatible-**

Highly compatible systems typically function well when other applications are running on a device. Compatibility also allows people who have different operating systems to use the same applications. For example, a compatible photo sharing application may offer the same features on an iOS device as it does on an Android device. You can determine compatibility for a particular application by reading the product description, which may include operating system information.

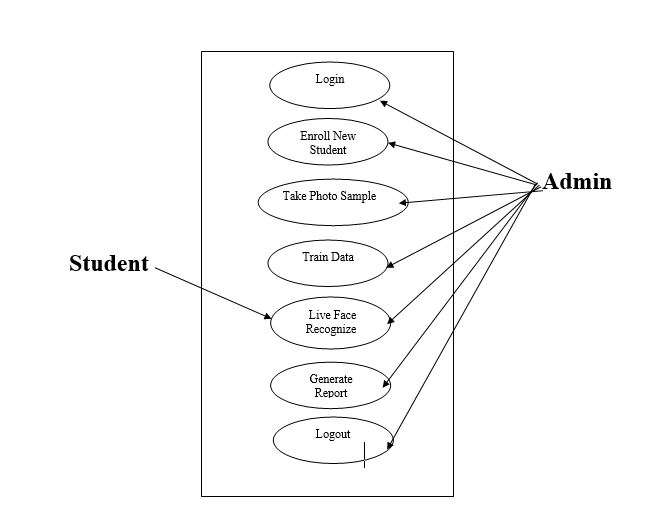
**Capacity -**

The capacity of a system refers to the amount of storage it offers. When using some applications, users can adjust and save settings based on their preferences.

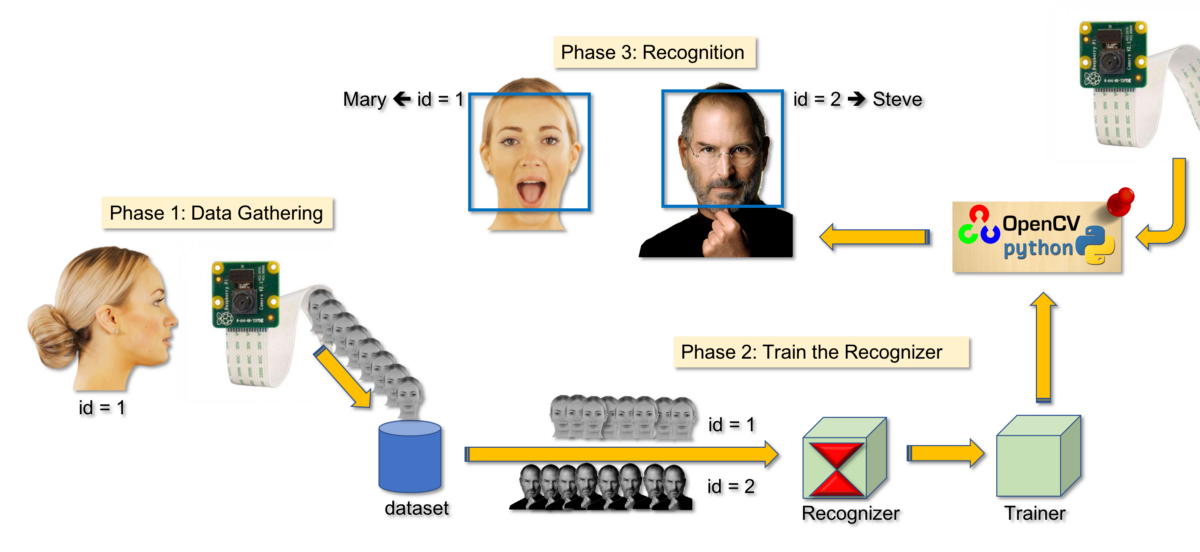
**CHAPTER - 4**

**SYSTEM DESIGN**

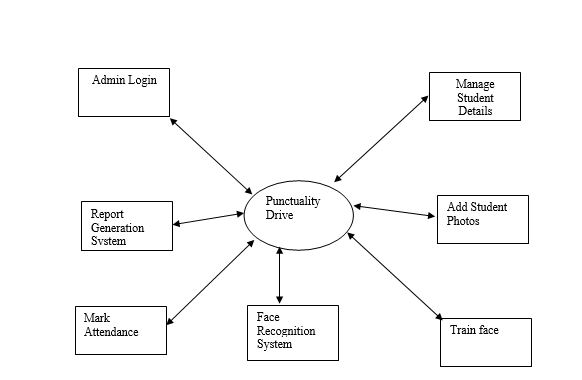
**4.1 Modular Design**



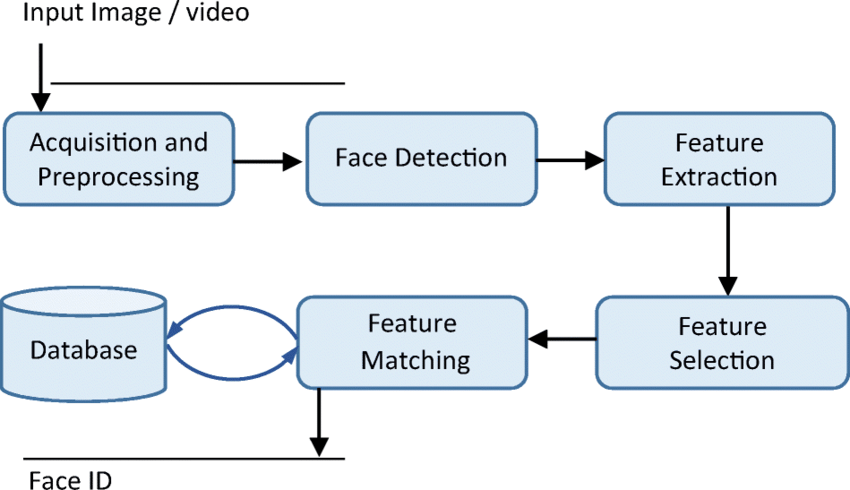
\



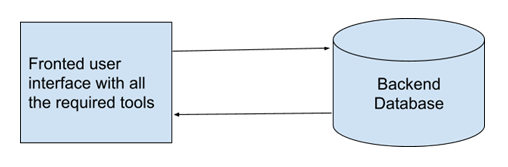
**4.2 Context Diagram**



**4.3 Structure Chart**



**4.4 File / Database Design**

****

**4.5 Normalization:-**

Normalization is the process of minimizing redundancy from a relation or set of relations. Redundancy in relation may cause insertion, deletion, and update anomalies. So, it helps to minimize the redundancy in relations. Normal forms are used to eliminate or reduce redundancy in database tables

**1. First Normal Form** –

If a relation contain composite or multi-valued attribute, it violates first normal form or a relation is in first normal form if it does not contain any composite or multi-valued attribute. A relation is in first normal form if every attribute in that relation is singled valued attribute.

**2. Second Normal Form –**

To be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency. A relation is in 2NF if it has No Partial Dependency, i.e., no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table.

**3. Third Normal Form –**

A relation is in third normal form, if there is no transitive dependency for non-prime attributes as well as it is in second normal form.

A relation is in 3NF if at least one of the following condition holds in every non-trivial function dependency X –> Y

**4. Boyce-Codd Normal Form (BCNF) –**

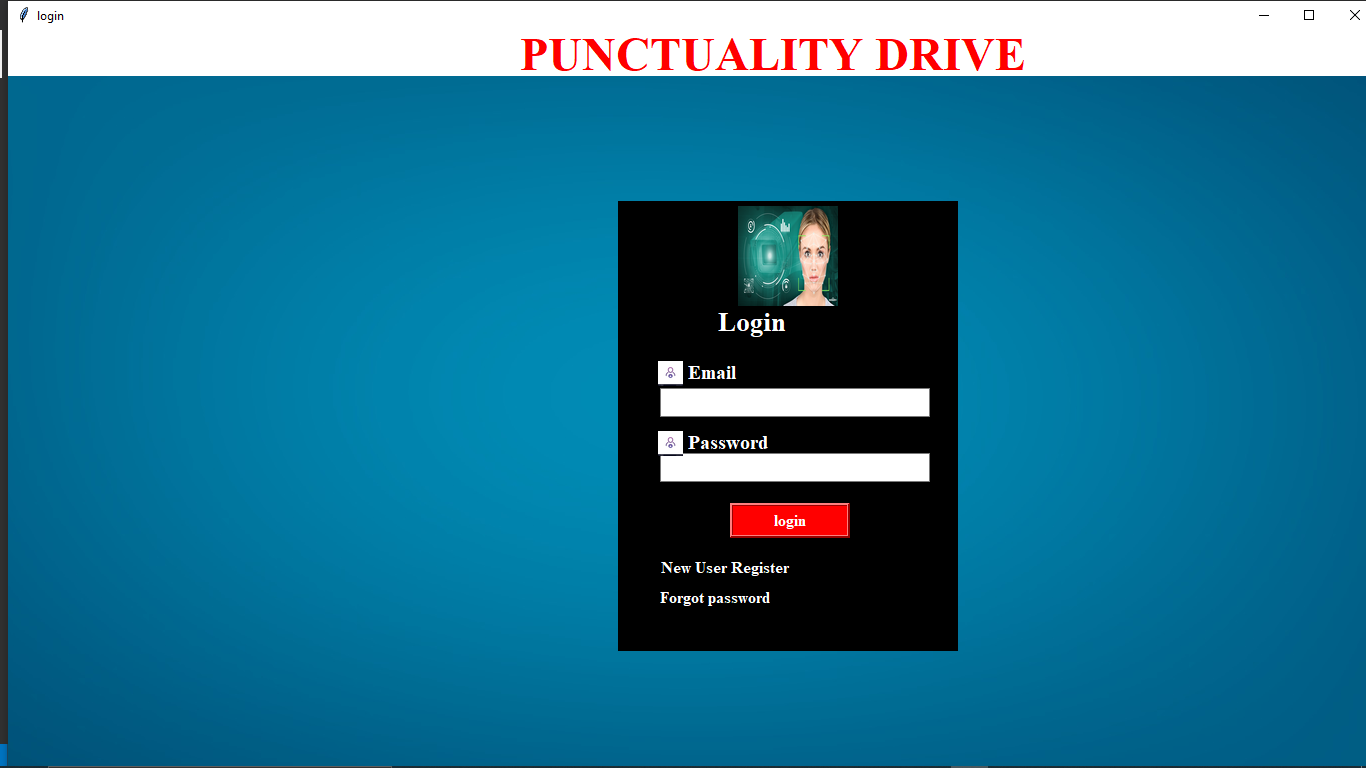
A relation R is in BCNF if R is in Third Normal Form and for every FD, LHS is super key. A relation is in BCNF if in every non-trivial functional dependency X –> Y, X is a super key.

**4.6 Input / Output form design:-**

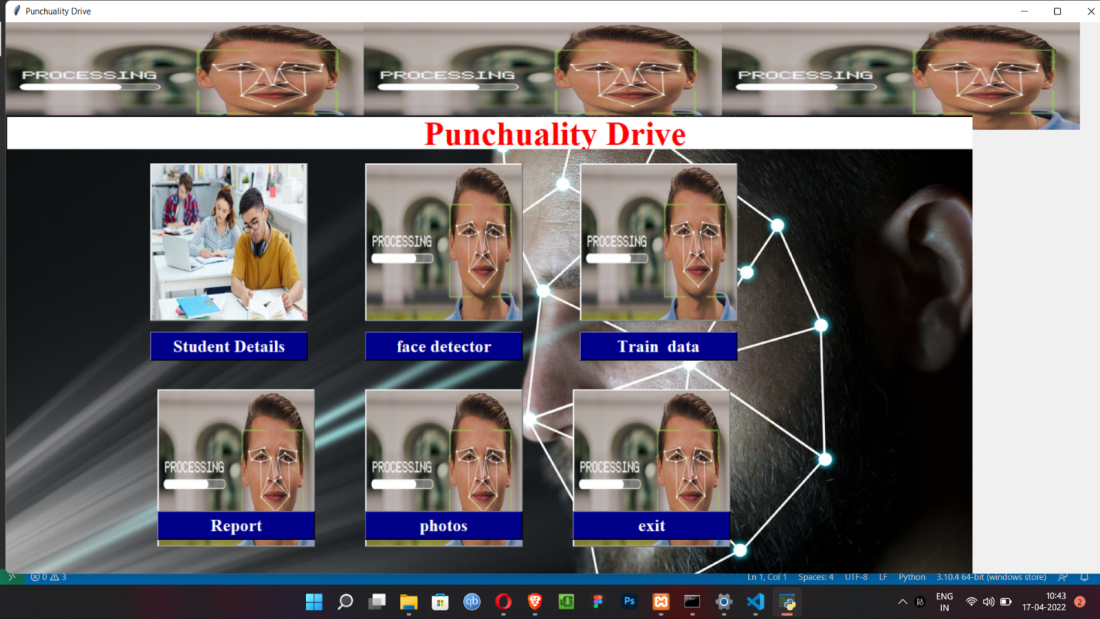
In an information system, input is the raw data that is processed to produce output.

* It should serve a specific purpose effectively such as storing, recording, and retrieving the information.
* It ensures proper completion with accuracy.
* It should be easy to fill and straightforward.
* It should focus on the user's attention, consistency, and simplicity.

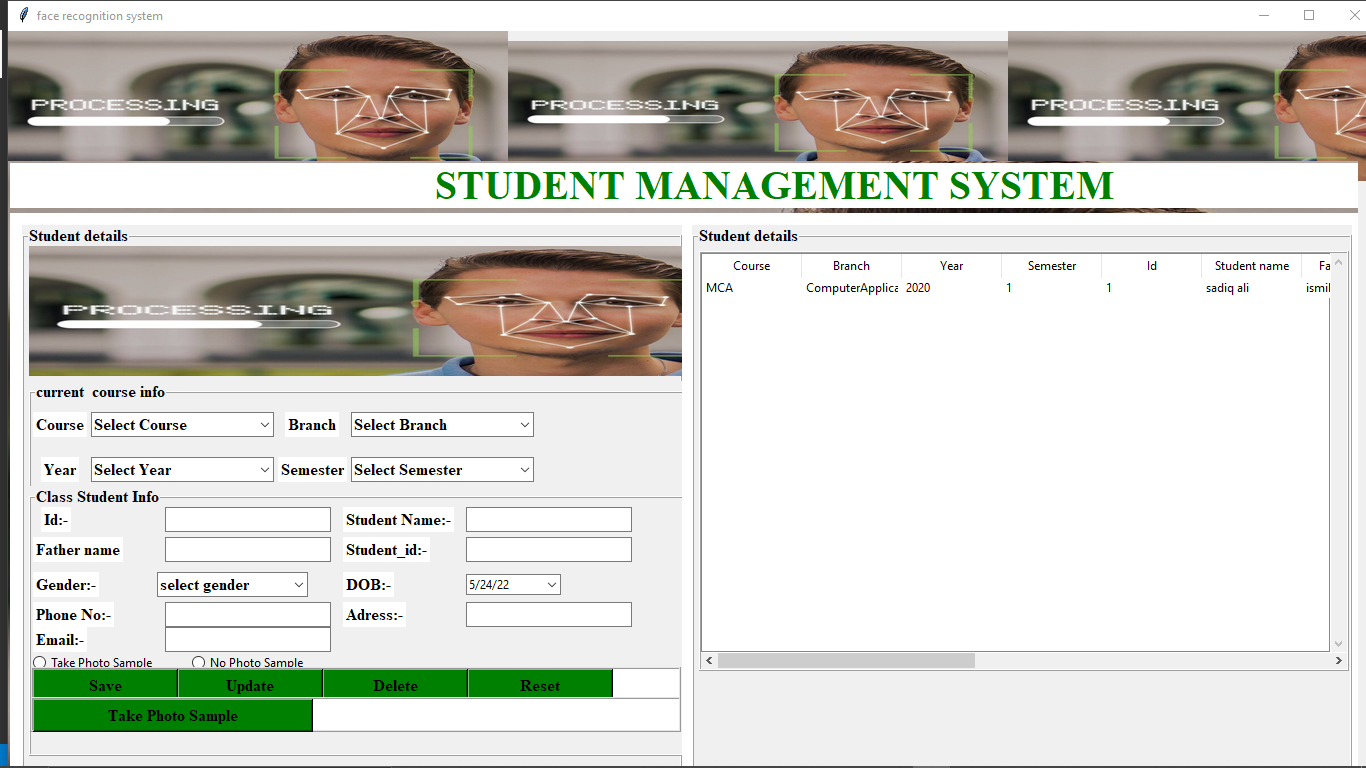
**Admin Login**



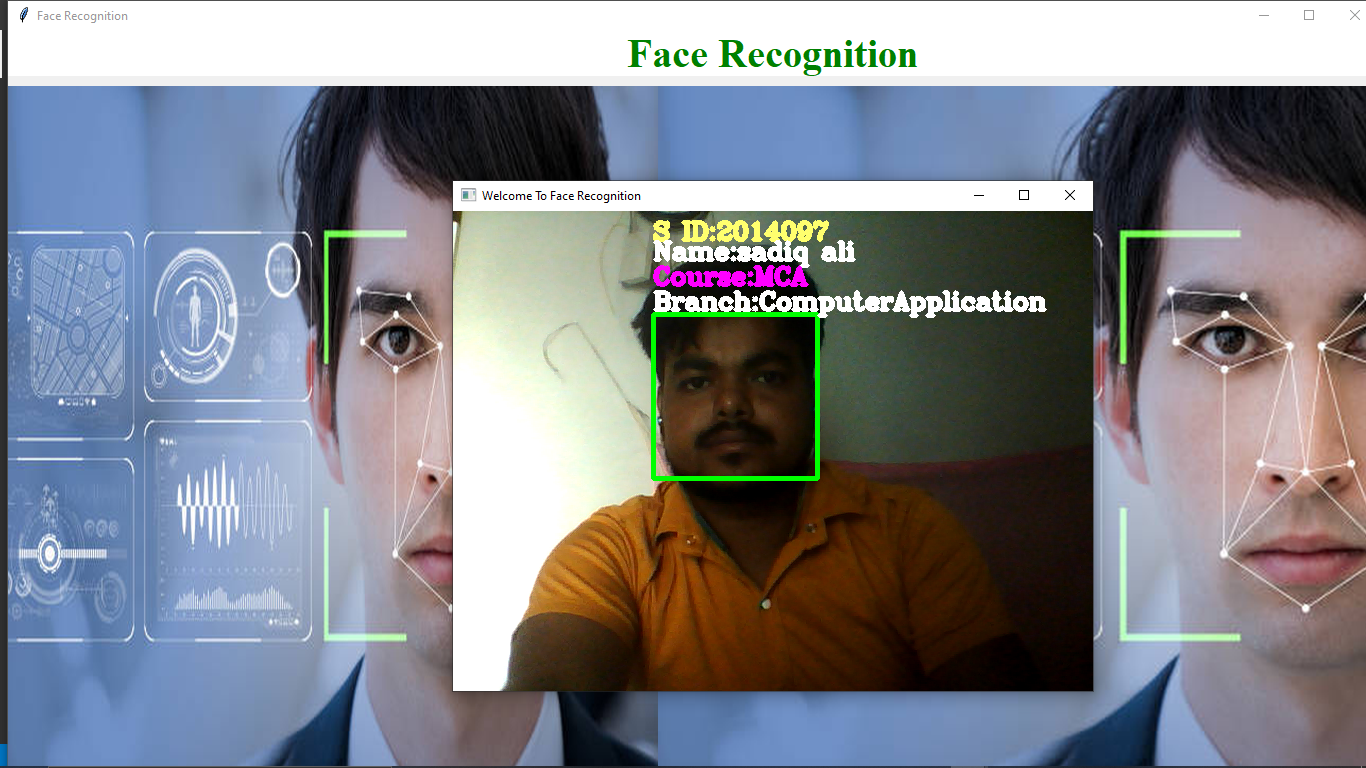
**Admin Dashboard**



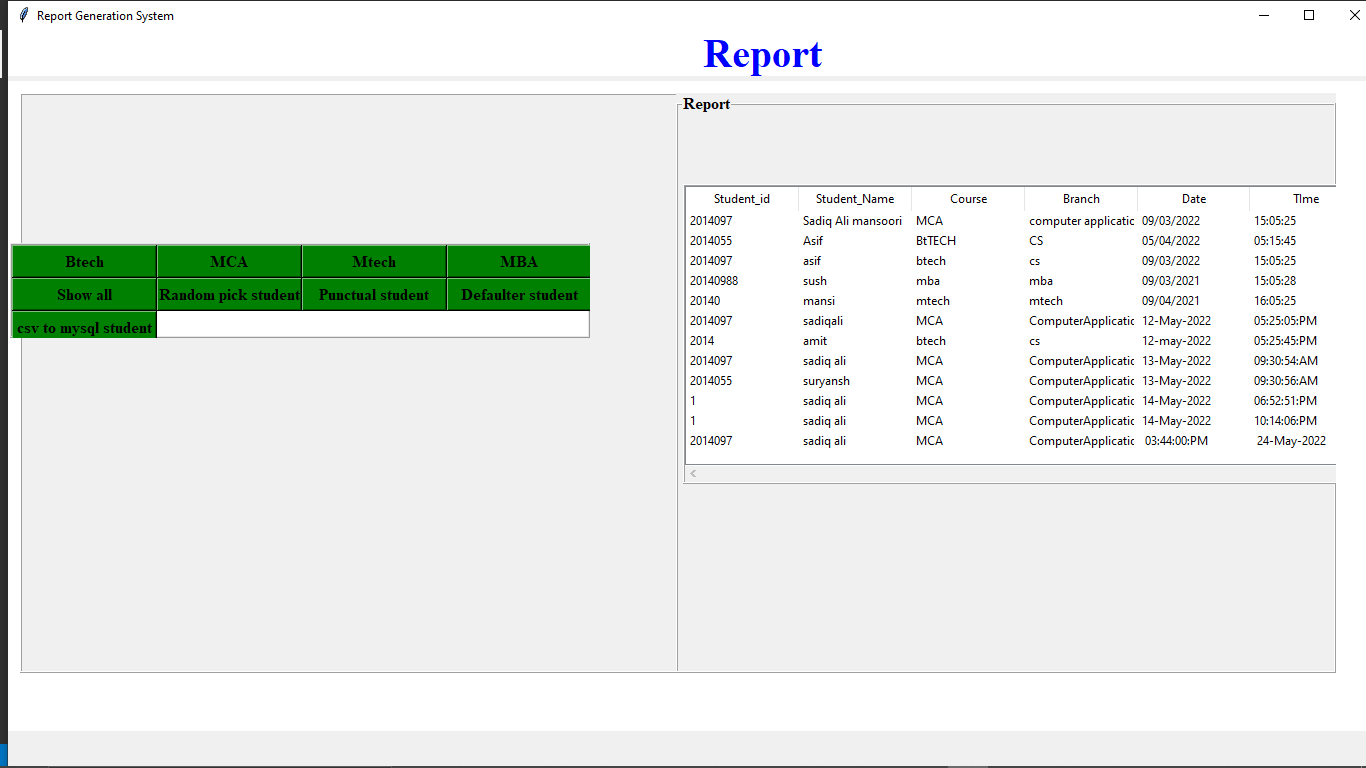
**Student Details**



Face Recognition



Report Generation system



**4.7 Screen Design**

****

**4.8 Report Design System**

In this logical system is built which fulfills the given requirements this phase of system requirements deals with transforming the clients requirements into a logically working system. Normally design is performed into following two steps.

**i. Primary Design :** In this phase the system is designed into block level. The blocks are created on the basis of the analysis done in the problem identification phase. Different blocks are created for different functions; emphasis is put on minimizing the information flow between the blocks. All activities which require more interaction are kept in one block.

1. **Secondary Design :** In this phase detailed design of every block is performed.

**CHAPTER - 5**

**SYSTEM IMPLEMENTATION**

**5.1 H/W and S/W Requirement:-**

**Hardware Requirement**

A desktop computer with Intel Core i3 64-bit processor and Graphic card 1 GB RAM, and Microsoft Windows 10 operating system was used.

**Software Requirement**

Front End: Tkinter Gui python

Backend: CSV file ,SQL

Harcascade classifier,LBPH Algorithm

**5.2 User Interface Design**

User Interface Design is concerned with the dialogue between a user and the computer. It is concerned with everything from starting the system or logging into the system to the eventual presentation of desired inputs and outputs. The overall flow of screens and messages is called a dialogue. The following steps are various guidelines for User Interface Design:

i. The system user should always be aware of what to do next.

1. The screen should be formatted so that various types of information, instructions and messages always appear in the same general display area.
2. Messages, instructions or information should be displayed long enough to allow the system user to read them.
3. Use display attributes sparingly.
4. Default values for fields and answers to be entered by the user should be specified.
5. A user should not be allowed to proceed without correcting an error.
6. The system user should never get an operating system message or fatal error.

5.3 Coding

* + 1. List of Programs

1. Main.py
2. Student.py
3. Train.py
4. Facerecog.py
5. Classifier.xml
6. Mark.csv
7. Finalreport.py

5.3.2 Inputs/ output

Login.py:-

from tkinter import\*

from tkinter import ttk

from PIL import Image,ImageTk

from tkinter import messagebox

import mysql.connector

import cv2

import os

import numpy as np

import csv

from tkinter import filedialog

from student import Student

import os

from report import Report

from facerecog import Face\_recognition

from train import Train

from register import Register

def main():

win=Tk()

app=Login\_window(win)

win.mainloop()

class Login\_window:

def \_\_init\_\_(self,root):

self.root=root

self.root.geometry("1550x800+0+0")

self.root.title("login")

self.bg=ImageTk.PhotoImage(file=r"D:\face recognition - Copy\college images\login.jpg")

lbl\_bg=Label(self.root,image=self.bg)

lbl\_bg.place(x=0,y=0,relwidth=1,relheight=1)

main\_frame=Frame(self.root,bd=2,bg="black")

main\_frame.place(x=610,y=170,width=340,height=450)

title\_lbl=Label(self.root,text="PUNCTUALITY DRIVE",font=("times new roman",35,"bold"),bg="white",fg="red")

title\_lbl.place(x=0,y=0,width=1530,height=45)

img1=Image.open(r"D:\face recognition - Copy\college images\detect.png")

img1=img1.resize((100,100),Image.ANTIALIAS)

self.pohotimage1=ImageTk.PhotoImage(img1)

img1\_lbl=Label(image=self.pohotimage1,bg="black",borderwidth=0)

img1\_lbl.place(x=730,y=175,width=100,height=100)

get\_str=Label(main\_frame,text="Login",font=("times new roman",20,"bold"),fg="white",bg="black")

get\_str.place(x=95,y=100)

# user name label

email=Label(main\_frame,text="Email",font=("times new roman",15,"bold"),fg="white",bg="black")

email.place(x=65,y=155)

# entry field user name

self.txtuser=ttk.Entry(main\_frame,font=("times new roman",15,"bold"))

self.txtuser.place(x=40,y=185,width=270)

#password

password=Label(main\_frame,text="Password",font=("times new roman",15,"bold"),fg="white",bg="black")

password.place(x=65,y=225)

# password entry

self.txtpass=ttk.Entry(main\_frame,show="\*",font=("times new roman",15,"bold"))

self.txtpass.place(x=40,y=250,width=270)

#### icon image

img2=Image.open(r"D:\face recognition - Copy\college images\userlogin.jpg")

img2=img2.resize((25,25),Image.ANTIALIAS)

self.pohotimage2=ImageTk.PhotoImage(img2)

img1\_lbl=Label(image=self.pohotimage2,bg="black",borderwidth=0)

img1\_lbl.place(x=650,y=330,width=25,height=25)

## second icon

img3=Image.open(r"D:\face recognition - Copy\college images\password.png")

img3=img3.resize((25,25),Image.ANTIALIAS)

self.pohotimage3=ImageTk.PhotoImage(img3)

img1\_lbl=Label(image=self.pohotimage2,bg="black",borderwidth=0)

img1\_lbl.place(x=650,y=400,width=25,height=25)

# button frame

login\_btn=Button(main\_frame,text="login",command=self.login,width=15,activebackground = "green",font=("times new roman",12,"bold"),bd=3,relief=RIDGE,fg="white",bg="red")

login\_btn.place(x=110,y=300,width=120,height=35)

# register button

register=Button(main\_frame,text="New User Register",command=self.registerwindow,width=15,activebackground = "black",font=("times new roman",12,"bold"),borderwidth=0,fg="white",bg="black")

register.place(x=25,y=350,width=160)

# forgot password

forgotpassword=Button(main\_frame,text="Forgot password",command=self.forgotpassword,width=15,activebackground = "black",font=("times new roman",12,"bold"),borderwidth=0,fg="white",bg="black")

forgotpassword.place(x=15,y=380,width=160)

# register window

def registerwindow(self):

self.new\_window=Toplevel(self.root)

self.app=Register(self.new\_window)

# login function

def login(self):

if self.txtuser.get()=="" or self.txtpass.get()=="":

messagebox.showerror("error","please enter user name and password")

else:

conn=mysql.connector.connect(host="localhost",user="root",password="",database="face\_recognizer")

my\_cursor=conn.cursor()

my\_cursor.execute("select \* from registration where Email=%s and Password=%s",(

self.txtuser.get(),

self.txtpass.get()

))

row=my\_cursor.fetchone()

if row==None:

messagebox.showerror("error","invalid username and password")

else:

open\_main=messagebox.askyesno("yes no","Access only admin")

if open\_main>0:

self.new\_window=Toplevel(self.root)

self.app=Face\_Recognition\_System(self.new\_window)

else:

if not open\_main:

return

conn.commit()

conn.close()

# signup function

def register(self):

if self.fname.get()=='':

messagebox.showerror("error","please enter first name")

elif self.Securityques.get()=='select':

messagebox.showerror("error","please choose security question ")

elif self.securityAnswer.get()=='':

messagebox.showerror("error","please enter security Answer ")

elif self.Password.get()!=self.Confirm\_password.get():

messagebox.showerror("error ","confirm password muust be same")

else: conn=mysql.connector.connect(host="localhost",user="root",password="",database="face\_recognizer")

my\_cursor=conn.cursor()

query=("select \* from registration where email=%s")

value=(self.Email.get(),)

my\_cursor.execute(query,value)

row=my\_cursor.fetchone()

if row!=None:

messagebox.showerror("error","user already exists,please try another email ")

else:

my\_cursor.execute("insert into registration values (%s,%s,%s,%s,%s,%s,%s)",(

self.fname.get(),

self.lname.get(), self.contact.get(), self.Email.get(), self.Securityques.get(), self.securityAnswer.get(), self.Confirm\_password.get(),))

conn.commit()

conn.close()

messagebox.showinfo("succes","Registration Succesfully ",parent=self.root)

if \_\_name\_\_=="\_\_main\_\_":

main()

**Student.py:-**

from tkinter  import\*

from tkinter import ttk

from tkinter import scrolledtext

from PIL import Image,ImageTk

# from main import Face\_Recognition\_System

from tkinter import messagebox

import mysql.connector

import cv2

import os

# from student import Student

class Student:

    def \_\_init\_\_(self,root):

        self.root=root

        self.root.geometry("1530x750+0+0")

        self.root.title("face recognition system")

        # varaibles

        self.var\_course=StringVar()

        self.var\_branch=StringVar()

        self.var\_year=StringVar()

        self.var\_sem=StringVar()

        self.var\_id=StringVar()

        self.var\_sname=StringVar()

        self.var\_fname=StringVar()

        self.var\_dob=StringVar()

        self.var\_gender=StringVar()

        self.var\_email=StringVar()

        self.var\_phone=StringVar()

        self.var\_adress=StringVar()

        self.var\_sid=StringVar()

        # self.var\_photo=StringVar()

        #first image

        img=Image.open("college images//detect.jpg")

        img=img.resize((500,150),Image.ANTIALIAS)

        self.photoimg=ImageTk.PhotoImage(img)

        f\_lbl=Label(self.root,image=self.photoimg)

        f\_lbl.place(x=0,y=0,width=500,height=150)

        #second image

        img1=Image.open("college images//detect.jpg")

        img1=img1.resize((500,130),Image.ANTIALIAS)

        self.photoimg1=ImageTk.PhotoImage(img1)

        f\_lbl=Label(self.root,image=self.photoimg1)

        f\_lbl.place(x=500,y=0,width=500,height=150)

#third image

        img2=Image.open("college images//detect.jpg")

        img2=img2.resize((500,150),Image.ANTIALIAS)

        self.photoimg2=ImageTk.PhotoImage(img2)

        f\_lbl=Label(self.root,image=self.photoimg2)

        f\_lbl.place(x=1000,y=0,width=500,height=150)

#bg image

        img3=Image.open("college images//detect.jpg")

        img3=img3.resize((1400,770),Image.ANTIALIAS)

        self.photoimg3=ImageTk.PhotoImage(img3)

        bg\_img=Label(self.root,image=self.photoimg3)

        bg\_img.place(x=0,y=130,width=1350,height=700)

        title\_lbl=Label(bg\_img,text="STUDENT MANAGEMENT SYSTEM",font=("times new roman",30,"bold"),bg="white",fg="green")

        title\_lbl.place(x=0,y=0,width=1530,height=45)

### home

        # home\_btn=Button(bg\_img,text="Home",command=self.home,width=12,font=("times new roman",12,"bold"),bg="green")

        # home\_btn.grid(row=0,column=3,padx=4)

#frame creating

        main\_frame=Frame(bg\_img,bd=2,bg="white")

        main\_frame.place(x=0,y=50,width=1530,height=650)

#left side label frame

        Left\_frame=LabelFrame(main\_frame,bd=2,relief=RIDGE,text="Student details",font=("times new roman",12,"bold"))

        Left\_frame.place(x=10,y=10,width=660,height=580)

#image

        img\_left=Image.open("college images//detect.jpg")

        img\_left=img\_left.resize((720,130),Image.ANTIALIAS)

        self.photoimg\_left=ImageTk.PhotoImage(img\_left)

        f\_lbl=Label(Left\_frame,image=self.photoimg\_left)

        f\_lbl.place(x=5,y=0,width=720,height=130)

        # current course

        current\_course\_frame=LabelFrame(Left\_frame,bd=2,relief=RIDGE,text="current  course info",font=("times new roman",12,"bold"))

        current\_course\_frame.place(x=5,y=135,width=720,height=140)

     # course

        course\_label=Label(current\_course\_frame,text="Course",font=("times new roman",12,"bold"),bg="white")

        course\_label.grid(row=0,column=0,padx=2,pady=10)

        course\_combo=ttk.Combobox(current\_course\_frame,textvariable=self.var\_course,font=("times new roman",12,"bold"),state="readonly",width=20)

        course\_combo["values"]=("Select Course","MCA","BTECH","MBA","MTECH")

        course\_combo.current(0)

        course\_combo.grid(row=0,column=1,padx=2,pady=10,sticky=W)

        #branch

        branch\_label=Label(current\_course\_frame,text="Branch",font=("times new roman",12,"bold"),bg="white")

        branch\_label.grid(row=0,column=2,padx=2,pady=10)

        branch\_combo=ttk.Combobox(current\_course\_frame,textvariable=self.var\_branch,font=("times new roman",12,"bold"),state="readonly")

        branch\_combo["values"]=("Select Branch","Computer Application","CSE","IT","CS/ML","CS/AI","CIVIL","ME","BIO TECH")

        branch\_combo.current(0)

        branch\_combo.grid(row=0,column=3,padx=2,pady=10)

#year

        year\_label=Label(current\_course\_frame,text="Year",font=("times new roman",12,"bold"),bg="white")

        year\_label.grid(row=1,column=0,padx=10,sticky=W)

        year\_combo=ttk.Combobox(current\_course\_frame,textvariable=self.var\_year,font=("times new roman",12,"bold"),state="readonly",width=20)

        year\_combo["values"]=("Select Year","2020","2021","2022")

        year\_combo.current(0)

        year\_combo.grid(row=1,column=1,padx=2,pady=10,sticky=W)

        #semester

        semester\_label=Label(current\_course\_frame,text="Semester",font=("times new roman",12,"bold"),bg="white")

        semester\_label.grid(row=1,column=2,padx=2,sticky=W)

        semester\_combo=ttk.Combobox(current\_course\_frame,textvariable=self.var\_sem,font=("times new roman",12,"bold"),state="readonly",width=20)

        semester\_combo["values"]=("Select Semester","1","2","3","4","5","6","7","8")

        semester\_combo.current(0)

        semester\_combo.grid(row=1,column=3,padx=2,pady=10,sticky=W)

         #  class student info

        class\_student\_frame=LabelFrame(Left\_frame,bd=2,relief=RIDGE,text="Class Student Info",font=("times new roman",12,"bold"))

        class\_student\_frame.place(x=5,y=240,width=720,height=270)

        #student id

        studentid\_label=Label(class\_student\_frame,text="Id:-",font=("times new roman",12,"bold"),bg="white")

        studentid\_label.grid(row=0,column=0,padx=10,sticky=W)

        Studentid\_entry=ttk.Entry(class\_student\_frame,validate="key",textvariable=self.var\_id,width=20,font=("times new roman",12,"bold"))

        Studentid\_entry.grid(row=0,column=1,padx=10,sticky=W)

        #student name

        student\_name\_label=Label(class\_student\_frame,text="Student Name:-",font=("times new roman",12,"bold"),bg="white")

        student\_name\_label.grid(row=0,column=2,padx=2,sticky=W)

        student\_name\_entry=ttk.Entry(class\_student\_frame,textvariable=self.var\_sname,width=20,font=("times new roman",12,"bold"))

        student\_name\_entry.grid(row=0,column=3,padx=10,sticky=W)

        # bind and validation

        # validate\_student\_name=self.root.register(self.checkname)

        # student\_name\_entry.config(validate='key',validatecommand=(validate\_student\_name,'%P'))

        #father name

        father\_name\_label=Label(class\_student\_frame,text="Father name",font=("times new roman",12,"bold"),bg="white")

        father\_name\_label.grid(row=1,column=0,padx=2,sticky=W)

        father\_name\_entry=ttk.Entry(class\_student\_frame,textvariable=self.var\_fname,width=20,font=("times new roman",12,"bold"))

        father\_name\_entry.grid(row=1,column=1,padx=10,pady=5,sticky=W)

        #gender

        Gender\_label=Label(class\_student\_frame,text="Gender:-",font=("times new roman",12,"bold"),bg="white")

        Gender\_label.grid(row=2,column=0,padx=2,sticky=W)

        # Gender\_entry=ttk.Entry(class\_student\_frame,textvariable=self.var\_gender,width=20,font=("times new roman",12,"bold"))

        # Gender\_entry.grid(row=2,column=1,padx=10,sticky=W)

        Gender\_combo=ttk.Combobox(class\_student\_frame,textvariable=self.var\_gender,font=("times new roman",12,"bold"),state="readonly",width=16)

        Gender\_combo["values"]=("select gender","Male","Female","Other")

        Gender\_combo.current(0)

        Gender\_combo.grid(row=2,column=1,padx=2,pady=5,sticky=W)

        #Email

        Email\_label=Label(class\_student\_frame,text="Email:-",font=("times new roman",12,"bold"),bg="white")

        Email\_label.grid(row=2,column=2,padx=2,sticky=W)

        Email\_entry=ttk.Entry(class\_student\_frame,textvariable=self.var\_email,width=20,font=("times new roman",12,"bold"))

        Email\_entry.grid(row=2,column=3,padx=10,sticky=W)

        #phone

        phoneno\_label=Label(class\_student\_frame,text="Phone No:-",font=("times new roman",12,"bold"),bg="white")

        phoneno\_label.grid(row=3,column=0,padx=2,sticky=W)

        phoneno\_entry=ttk.Entry(class\_student\_frame,textvariable=self.var\_phone,width=20,font=("times new roman",12,"bold"))

        phoneno\_entry.grid(row=3,column=1,padx=10,sticky=W)

        # validate

        validate\_phoneno=self.root.register(self.checkcontact)

        phoneno\_entry.config(validate='key',validatecommand=(validate\_phoneno,'%P'))

        #adress

        Adress\_label=Label(class\_student\_frame,text="Adress:-",font=("times new roman",12,"bold"),bg="white")

        Adress\_label.grid(row=3,column=2,padx=2,sticky=W)

        Adress\_entry=ttk.Entry(class\_student\_frame,textvariable=self.var\_adress,width=20,font=("times new roman",12,"bold"))

        Adress\_entry.grid(row=3,column=3,padx=10,sticky=W)

        #DOB

        DOB\_label=Label(class\_student\_frame,text="DOB:-",font=("times new roman",12,"bold"),bg="white")

        DOB\_label.grid(row=4,column=0,padx=2,sticky=W)

        DOB\_entry=ttk.Entry(class\_student\_frame,textvariable=self.var\_dob,width=20,font=("times new roman",12,"bold"))

        DOB\_entry.grid(row=4,column=1,padx=10,sticky=W)

        #sudent id

        student\_id\_label=Label(class\_student\_frame,text="Student\_id:-",font=("times new roman",12,"bold"),bg="white")

        student\_id\_label.grid(row=1,column=2,padx=2,sticky=W)

        student\_id\_entry=ttk.Entry(class\_student\_frame,textvariable=self.var\_sid,width=20,font=("times new roman",12,"bold"))

        student\_id\_entry.grid(row=1,column=3,padx=10,sticky=W)

        #radio buttons

        self.var\_radio1=StringVar()

        radiobtn1=ttk.Radiobutton(class\_student\_frame,variable=self.var\_radio1,text="Take Photo Sample",value="Yes")

        radiobtn1.grid(row=5,column=0)

        radiobtn2=ttk.Radiobutton(class\_student\_frame,variable=self.var\_radio1,text="No Photo Sample",value="No")

        radiobtn2.grid(row=5,column=1)

        #button frame

        btn\_frame=Frame(class\_student\_frame,bd=2,relief=RIDGE,bg="white")

        btn\_frame.place(x=0,y=160,width=650,height=35)

        #save button

        save\_btn=Button(btn\_frame,text="Save",command=self.add\_data,width=15,activebackground = "Red",font=("times new roman",12,"bold"),bg="green")

        save\_btn.grid(row=0,column=0)

        #update

        update\_btn=Button(btn\_frame,text="Update",command=self.update\_data,width=15,activebackground = "Red",font=("times new roman",12,"bold"),bg="green")

        update\_btn.grid(row=0,column=1)

        #delete

        delete\_btn=Button(btn\_frame,text="Delete",command=self.delete\_data,width=15,activebackground = "Red",font=("times new roman",12,"bold"),bg="green")

        delete\_btn.grid(row=0,column=2)

        #reset

        reset\_btn=Button(btn\_frame,text="Reset",command=self.reset\_data,width=15,activebackground = "Red",font=("times new roman",12,"bold"),bg="green")

        reset\_btn.grid(row=0,column=3)

        #button frame1

        btn\_frame1=Frame(class\_student\_frame,bd=2,relief=RIDGE,bg="white")

        btn\_frame1.place(x=0,y=190,width=650,height=35)

        #take photo sample

        takephoto\_btn=Button(btn\_frame1,command=self.generate\_dataset,text="Take Photo Sample",activebackground = "Red",width=30,font=("times new roman",12,"bold"),bg="green")

        takephoto\_btn.grid(row=0,column=0)

        # update photo sample

        # takephoto\_btn=Button(btn\_frame1,text="Update photo sample",width=30,activebackground = "Red",font=("times new roman",12,"bold"),bg="green")

        # takephoto\_btn.grid(row=0,column=1)

        #right side label frame

        right\_frame=LabelFrame(main\_frame,bd=2,relief=RIDGE,text="Student details",font=("times new roman",12,"bold"))

        right\_frame.place(x=680,y=10,width=660,height=580)

        # table\_frame

        table\_frame=Frame(right\_frame,bd=2,relief=RIDGE)

        # table\_frame.place(x=5,y=210,width=700,height=200)

        table\_frame.place(x=5,y=5,width=650,height=420)

        scroll\_x=ttk.Scrollbar(table\_frame,orient=HORIZONTAL)

        scroll\_y=ttk.Scrollbar(table\_frame,orient=VERTICAL)

        self.student\_table=ttk.Treeview(table\_frame,column=("course","branch","year","sem","id","sname","fname","Student id","gender","email","phone","adress","dob","photo"),xscrollcommand=scroll\_x.set,yscrollcommand=scroll\_y.set)

        scroll\_x.pack(side=BOTTOM,fill=X)

        scroll\_y.pack(side=RIGHT,fill=Y)

        scroll\_x.config(command=self.student\_table.xview)

        scroll\_y.config(command=self.student\_table.yview)

        self.student\_table.heading("course",text="Course")

        self.student\_table.heading("branch",text="Branch")

        self.student\_table.heading("year",text="Year")

        self.student\_table.heading("sem",text="Semester")

        self.student\_table.heading("id",text="Id")

        self.student\_table.heading("sname",text="Student name")

        self.student\_table.heading("fname",text="Father name")

        self.student\_table.heading("Student id",text="Student id")

        self.student\_table.heading("gender",text="Gender")

        self.student\_table.heading("email",text="Email")

        self.student\_table.heading("phone",text="Phone")

        self.student\_table.heading("adress",text="Adress")

        self.student\_table.heading("dob",text="Date of birth")

        self.student\_table.heading("photo",text="PhotoSampleStatus")

        self.student\_table["show"]="headings"

        self.student\_table.column("course",width=100)

        self.student\_table.column("branch",width=100)

        self.student\_table.column("year",width=100)

        self.student\_table.column("sem",width=100)

        self.student\_table.column("id",width=100)

        self.student\_table.column("sname",width=100)

        self.student\_table.column("fname",width=100)

        self.student\_table.column("dob",width=100)

        self.student\_table.column("gender",width=100)

        self.student\_table.column("email",width=100)

        self.student\_table.column("phone",width=100)

        self.student\_table.column("adress",width=100)

        self.student\_table.column("photo",width=150)

        self.student\_table.pack(fill=BOTH,expand=1)

        self.student\_table.bind("<ButtonRelease>",self.get\_cursor)

        self.fetch\_data()

       # function declaration

    def add\_data(self):

            if self.var\_course.get()=="select course":

                messagebox.showerror("Error","select course",parent=self.root)

          elif self.var\_sid.get()=="":

                messagebox.showerror("Error","enter Syudent\_id",parent=self.root)

             else:                 conn=mysql.connector.connect(host="localhost",user="root",password="",database="face\_recognizer")

                    my\_cursor=conn.cursor()

                    my\_cursor.execute("insert into studentdetails values (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",(

                                                                                                    self.var\_course.get(),

                                                                                                    self.var\_branch.get(),

                                                                                                    self.var\_year.get(),

                                                                                                    self.var\_sem.get(),

                                                                                                    self.var\_id.get(),

                                                                                                    self.var\_sname.get(),

                                                                                                    self.var\_fname.get(),

                                                                                                    self.var\_sid.get(),

                                                                                                    self.var\_gender.get(),

                                                                                                    self.var\_email.get(),

                                                                                                    self.var\_phone.get(),

                                                                                                    self.var\_adress.get(),

                                                                                                    self.var\_dob.get(),

                                                                                                    self.var\_radio1.get()

                                                                                                    ))

                    conn.commit()

                    self.fetch\_data()

                    conn.close()

                    messagebox.showinfo("succes","student details are succesfully added",parent=self.root)

                # except Exception as es:

                # # messagebox.showerror("error",f"due to :{str(es)}",parent=self.root)

    # fetch data

    def fetch\_data(self):

        conn=mysql.connector.connect(host="localhost",user="root",password="",database="face\_recognizer")

        my\_cursor=conn.cursor()

        my\_cursor.execute("select \* from studentdetails")

        data=my\_cursor.fetchall()

        if len(data)!=0:

            self.student\_table.delete(\*self.student\_table.get\_children())

            for i in data :

                self.student\_table.insert("",END,values=i)

                conn.commit()

            conn.close()

         # ========get cursor==

    def get\_cursor(self,event=""):

        cursor\_focus=self.student\_table.focus()

        content=self.student\_table.item(cursor\_focus)

        data=content["values"]

        self.var\_course.set(data[0]),

        self.var\_branch.set(data[1]),

        self.var\_year.set(data[2]),

        self.var\_sem.set(data[3]),

        self.var\_id.set(data[4]),

        self.var\_sname.set(data[5]),

        self.var\_fname.set(data[6]),

        self.var\_sid.set(data[7]),

        self.var\_gender.set(data[8]),

        self.var\_email.set(data[9]),

        self.var\_phone.set(data[10]),

        self.var\_adress.set(data[11]),

        self.var\_dob.set(data[12]),

        self.var\_radio1.set(data[13])

    # update function

    def update\_data(self):

        if self.var\_course.get()=="select course" or self.var\_sname.get()=="" or self.var\_sid.get()=="":

            messagebox.showerror("Error","all field are required",parent=self.root)

        else:

            upadate=messagebox.askyesno("update","do u want to update this student details",parent=self.root)

            if upadate>0:

                conn=mysql.connector.connect(host="localhost",user="root",password="",database="face\_recognizer")

                my\_cursor=conn.cursor()

                my\_cursor.execute("update studentdetails set Course = %s, Branch = %s, Year = %s, Sem = %s,, Student\_Name = %s, Father\_Name = %s,Student\_Id=%s, Gender = %s, Email = %s, Phone = %s, Adress = %s,Dob = %s, PhotoSample = %s where id=%s "(

                                                                                                                                        self.var\_course.get(),

                                                                                                                                        self.var\_branch.get(),

                                                                                                                                        self.var\_year.get(),

                                                                                                                                        self.var\_sem.get(),

self.var\_sname.get(),

                                                                                                                                        self.var\_fname.get(),

                                                                                                                                        self.var\_sid.get(),

                                                                                                                                        self.var\_gender.get(),

                                                                                                                                        self.var\_email.get(),

                                                                                                                                        self.var\_phone.get(),

                                                                                                                                        self.var\_adress.get(),

                                                                                                                                        self.var\_dob.get(),

                                                                                                                                        self.var\_radio1.get(),

self.var\_id.get()  ))

            else:

                if not upadate:

                    return

            messagebox.showinfo("success","Student details succesfully update succesfully",parent=self.root)

            conn.commit()

            self.fetch\_data()

            conn.close()

      #delete function

    def delete\_data(self):

        if self.var\_sid.get()=="":

            messagebox.showerror("error","student id must required",parent=self.root)

        else:

            delete=messagebox.askyesno("student delete page","do u want to delete this student details",parent=self.root)

        if delete>0:                conn=mysql.connector.connect(host="localhost",user="root",password="",database="face\_recognizer")

                my\_cursor=conn.cursor()

                sql="delete from studentdetails where Student\_id=%s"

                val=(self.var\_sid.get(),)

                my\_cursor.execute(sql,val)

            else:

                if not delete:

                    return

            conn.commit()

            self.fetch\_data()

            conn.close()

            messagebox.showinfo("Delete ","Succesfully  Deleted Student Details",parent=self.root)

    # generate data set or take photo samples

    def generate\_dataset(self):

        if self.var\_course.get()=="select course" or self.var\_sname.get()=="" or self.var\_sid.get()=="":

            messagebox.showerror("Error","all field are required",parent=self.root)

        else:

            conn=mysql.connector.connect(host="localhost",user="root",password="",database="face\_recognizer")

            my\_cursor=conn.cursor()

            my\_cursor.execute("select \* from studentdetails")

            myresult=my\_cursor.fetchall()

            id=0

            for x in myresult:

                id+=1

            my\_cursor.execute("update studentdetails set Course = %s, Branch = %s, Year = %s, Sem = %s, id=%s,Student\_Name = %s, Father\_Name = %s,Student\_Id=%s, Gender = %s, Email = %s, Phone = %s, Adress = %s,Dob = %s, PhotoSample = %s where id=%s",(

                                                                                                    self.var\_course.get(),

                                                                                                    self.var\_branch.get(),

                                                                                                    self.var\_year.get(),

                                                                                                    self.var\_sem.get(),

                                                                                                    self.var\_id.get(),

                                                                                                    self.var\_sname.get(),

                                                                                                    self.var\_fname.get(),

                                                                                                    self.var\_sid.get(),

                                                                                                    self.var\_gender.get(),

                                                                                                    self.var\_email.get(),

                                                                                                    self.var\_phone.get(),

                                                                                                    self.var\_adress.get(),

                                                                                                    self.var\_dob.get(),

                                                                                                    self.var\_radio1.get(),

                                                                                                    self.var\_sid.get()==id+1

                                                                             ))

            conn.commit()

            self.fetch\_data()

            self.reset\_data()

            conn.close()

                    # ===== load predefined data on face frontal from opencv

            face\_classifier=cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml")

            def face\_cropped(img):

                gray=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

                faces=face\_classifier.detectMultiScale(gray,1.3,5)

                #scaling factor

                #minimum neighbour

                for (x,y,w,h) in faces:

                    face\_cropped=img[y:y+h,x:x+w]

                    return face\_cropped

            cap=cv2.VideoCapture(0)

            img\_id=0

            while True:

                ret,my\_frame=cap.read()

                if face\_cropped(my\_frame) is not None:

                    img\_id+=1

                    face=cv2.resize(face\_cropped(my\_frame),(550,550))

                    face=cv2.cvtColor(face,cv2.COLOR\_BGR2GRAY)

                    file\_name\_path="data/user."+str(id)+"."+str(img\_id)+".jpg"

                    cv2.imwrite(file\_name\_path,face)

                    cv2.putText(face,str(img\_id),(50,50),cv2.FONT\_HERSHEY\_COMPLEX,2,(0,255,0),2)

                    cv2.imshow("cropped face",face)

                if cv2.waitKey(1)==13 or int(img\_id)==50:

                    break

            cap.release()

            cv2.destroyAllWindows()

            messagebox.showinfo("result","generating data set succesfully")

    # def  home(self):

    #     self.new\_window=Toplevel(self.root)

    #     self.app=Face\_Recognition\_System(self.new\_window)

if \_\_name\_\_=="\_\_main\_\_":

    root=Tk()

    obj=Student(root)

    root.mainloop()

Train.py:-

from tkinter  import\*

from tkinter import ttk

from tkinter import scrolledtext

from PIL import Image,ImageTk

from tkinter import messagebox

import mysql.connector

import cv2

import os

import numpy as np

class Train:

    def \_\_init\_\_(self,root):

        self.root=root

        self.root.geometry("1530x790+0+0")

        self.root.title("face recognition system")

        title\_lbl=Label(self.root,text="Train Data",font=("times new roman",30,"bold"),bg="white",fg="BLACK")

        title\_lbl.place(x=0,y=0,width=1530,height=45)

        img\_top=Image.open( "college images//detect.jpg")

        img\_top=img\_top.resize((1530,325),Image.ANTIALIAS)

        self.photoimg\_top=ImageTk.PhotoImage(img\_top)

        f\_lbl=Label(self.root,image=self.photoimg\_top)

        f\_lbl.place(x=0,y=55,width=1530,height=325)

        # button

        b1\_1=Button(self.root,text="Trian Data",command=self.train\_classifier,cursor="hand2",font=("times new roman",18,"bold"),bg="darkblue",fg="white")

        b1\_1.place(x=0,y=380,width=1530,height=60)

        # second image

        img\_top=Image.open( "college images//detect.jpg")

        img\_top=img\_top.resize((1530,325),Image.ANTIALIAS)

        self.photoimg\_right=ImageTk.PhotoImage(img\_top)

        f\_lbl=Label(self.root,image=self.photoimg\_right)

        f\_lbl.place(x=0,y=440,width=1530,height=320)

    def train\_classifier(self):

        data\_path="data"

        path=[os.path.join(data\_path,file) for file in os.listdir(data\_path)]

        face=[]

        ids=[]

        for image in path:

            img=Image.open(image).convert('L')    # gray scale image

            imageNp=np.array(img,'uint8')

            id=int(os.path.split(image)[1].split('.')[1])

            face.append(imageNp)

            ids.append(id)

            cv2.imshow("Training",imageNp)

            cv2.waitKey(1)==13

        ids=np.array(ids)

        # print(ids)

        # print(face)

        ### train classifier and save

        clf=cv2.face.LBPHFaceRecognizer\_create()

        clf.train(face,ids)

        clf.write("classifier.xml")

        cv2.destroyAllWindows()

        messagebox.showinfo("result","train datas set successfully")

if \_\_name\_\_=="\_\_main\_\_":

    root=Tk()

    obj=Train(root)

    root.mainloop()

**Facerecog.py:-**

import cv2

import mysql.connector

from tkinter import \*

from PIL import Image, ImageTk

from tkinter import ttk

from tkinter import messagebox

from time import strftime

from datetime import datetime

import os

class Face\_recognition:

def \_\_init\_\_(self,root):

self.root=root

self.root.geometry("1530x790+0+0")

self.root.title("Face Recognition ")

title\_lbl=Label(self.root,text="Face Recognition",font=("times new roman",30,"bold"),bg="white",fg="green")

title\_lbl.place(x=0,y=0,width=1530,height=45)

img\_top=Image.open( "college images//facerecog.jpg")

img\_top=img\_top.resize((650,700),Image.ANTIALIAS)

self.photoimg\_top=ImageTk.PhotoImage(img\_top)

f\_lbl=Label(self.root,image=self.photoimg\_top)

f\_lbl.place(x=0,y=55,width=650,height=700)

# second image

img\_bottom=Image.open( "college images//facerecog.jpg")

img\_bottom=img\_bottom.resize((950,700),Image.ANTIALIAS)

self.photoimg\_bottom=ImageTk.PhotoImage(img\_bottom)\

f\_lbl=Label(self.root,image=self.photoimg\_bottom)

f\_lbl.place(x=650,y=55,width=950,height=700)

#button

b1\_1=Button(f\_lbl,text="Face Recognition",cursor="hand2",command=self.face\_recog,font=("times new roman",18,"bold"),bg="darkblue",fg="white")

b1\_1.place(x=200,y=450,width=200,height=40)

def mark\_attendance(self,i,n,c,b):

with open('mark.csv','r+') as f:

myDataList = f.readlines()

nameList = []

for line in myDataList:

entry = line.split(',')

nameList.append(entry[0])

if ((i not in nameList) and (n not in nameList)and (c not in nameList)and (b not in nameList)):

now = datetime.now()

time = now.strftime('%I:%M:%S:%p')

date = now.strftime('%d-%B-%Y')

f.writelines(f"\n{i},{n},{c},{b},{date},{time}")

def face\_recog(self):

def draw\_boundray(img,classifier,scaleFactor,minNeighbors,color,text,clf):

gray\_image=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

features=classifier.detectMultiScale(gray\_image,scaleFactor,minNeighbors)

coord=[]

for (x,y,w,h) in features:

cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),3)

id,predict=clf.predict(gray\_image[y:y+h,x:x+w])

confidence=int((100\*(1-predict/300)))

conn=mysql.connector.connect(host="localhost",user="root",password="",database="face\_recognizer")

my\_cursor=conn.cursor()

my\_cursor.execute("select Student\_Id from studentdetails where id="+str(id))

i=my\_cursor.fetchone()

i=''.join(i)

my\_cursor.execute("select Student\_Name from studentdetails where id="+str(id))

n=my\_cursor.fetchone()

n=''.join(n)

my\_cursor.execute("select Course from studentdetails where id="+str(id))

c=my\_cursor.fetchone()

c=''.join(c)

my\_cursor.execute("select Branch from studentdetails where id="+str(id))

b=my\_cursor.fetchone()

b=''.join(b)

if confidence>75:

cv2.putText(img, f"S ID:{i}",(x,y-75),cv2.FONT\_HERSHEY\_COMPLEX,0.8,(110,255,255),3)

cv2.putText(img, f"Name:{n}",(x,y-55),cv2.FONT\_HERSHEY\_COMPLEX,0.8,(255,255,255),3)

cv2.putText(img, f"Course:{c}",(x,y-30),cv2.FONT\_HERSHEY\_COMPLEX,0.8,(255,0,255),3)

cv2.putText(img, f"Branch:{b}",(x,y-5),cv2.FONT\_HERSHEY\_COMPLEX,0.8,(255,255,255),3)

self.mark\_attendance(i,n,c,b)

# self.mark\_attendance(i,n,)

else:

cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),3)

cv2.putText(img,"Unknown Face",(x,y-55),cv2.FONT\_HERSHEY\_COMPLEX,0.8,(255,255,255),3)

coord=[x,y,w,h]

return coord

def recognize(img,clf,faceCascade):

coord=draw\_boundray(img,faceCascade,1.1,10,(255,25,255),"Face",clf)

return img

faceCascade=cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml")

clf=cv2.face.LBPHFaceRecognizer\_create()

clf.read("classifier.xml")

video\_cap=cv2.VideoCapture(0)

while True:

ret,img=video\_cap.read()

img=recognize(img,clf,faceCascade)

cv2.imshow("Welcome To Face Recognition",img)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

video\_cap.release()

cv2.destroyAllWindows()

if \_\_name\_\_=="\_\_main\_\_":

root=Tk()

obj=Face\_recognition(root)

root.mainloop()

**Finalreport.py:-**

from tkinter import\*

from tkinter import ttk

from tkinter import scrolledtext

from PIL import Image,ImageTk

from tkinter import messagebox

import mysql.connector

import cv2

import os

import numpy as np

from tkinter import filedialog

class Report:

def \_\_init\_\_(self,root):

self.root=root

self.root.geometry("1530x790+0+0")

self.root.title("Report Generation System")

title\_lbl=Label(self.root,text="Report ",font=("times new roman",30,"bold"),bg="white",fg="Blue")

title\_lbl.place(x=0,y=0,width=1530,height=45)

# MAIN FRAME

main\_frame=Frame(self.root,bd=2,bg="white")

main\_frame.place(x=0,y=50,width=1530,height=650)

# LEFT FRAME

Left\_frame=LabelFrame(main\_frame,bd=2,relief=RIDGE,font=("times new roman",12,"bold"))

Left\_frame.place(x=10,y=10,width=660,height=580)

# frame

btn\_frame=Frame(main\_frame,bd=2,relief=RIDGE,bg="white")

btn\_frame.place(x=0,y=160,width=580,height=95)

#save button

Btech\_btn=Button(btn\_frame,text="Btech",command=self.btech,width=15,font=("times new roman",12,"bold"),bg="green")

Btech\_btn.grid(row=0,column=0)

#update

MCA\_btn=Button(btn\_frame,text="MCA",command=self.mca,width=15,font=("times new roman",12,"bold"),bg="green")

MCA\_btn.grid(row=0,column=1)

#delete

Mtech\_btn=Button(btn\_frame,text="Mtech",command=self.mtech,width=15,font=("times new roman",12,"bold"),bg="green")

Mtech\_btn.grid(row=0,column=2)

#reset

MBA\_btn=Button(btn\_frame,text="MBA",command=self.mba,width=15,font=("times new roman",12,"bold"),bg="green")

MBA\_btn.grid(row=0,column=3)

# export\_csv\_btn=Button(btn\_frame,text="Export csv",width=15,command=self.exportCsv,font=("times new roman",12,"bold"),bg="green")

# export\_csv\_btn.grid(row=1,column=1)

Showall\_csv\_btn=Button(btn\_frame,text="Show all",width=15,command=self.showall,font=("times new roman",12,"bold"),bg="green")

Showall\_csv\_btn.grid(row=1,column=0)

random\_csv\_btn=Button(btn\_frame,text="Random pick student",width=15,command=self.random,font=("times new roman",12,"bold"),bg="green")

random\_csv\_btn.grid(row=1,column=1)

Punctual\_btn=Button(btn\_frame,text="Punctual student",width=15,command=self.punctual,font=("times new roman",12,"bold"),bg="green")

Punctual\_btn.grid(row=1,column=2)

Defaulter\_btn=Button(btn\_frame,text="Defaulter student",width=15,command=self.defaulter,font=("times new roman",12,"bold"),bg="green")

Defaulter\_btn.grid(row=1,column=3)

csvtomysql\_btn=Button(btn\_frame,text="csv to mysql student",width=15,command=self.csv,font=("times new roman",12,"bold"),bg="green")

csvtomysql\_btn.grid(row=2,column=0)

# button frame1

# btn\_frame1=Frame(class\_student\_frame,bd=2,relief=RIDGE,bg="white")

# btn\_frame1.place(x=0,y=190,width=650,height=35)

# #take photo sample

# takephoto\_btn=Button(btn\_frame1,text="Take Photo Sample",width=30,font=("times new roman",12,"bold"),bg="green")

# takephoto\_btn.grid(row=0,column=0)

# # update photo sample

# takephoto\_btn=Button(btn\_frame1,text="Update photo sample",width=30,font=("times new roman",12,"bold"),bg="green")

# takephoto\_btn.grid(row=0,column=1)

# right frame

right\_frame=LabelFrame(main\_frame,bd=2,relief=RIDGE,text="Report",font=("times new roman",12,"bold"))

right\_frame.place(x=666,y=10,width=660,height=580)

table\_frame=Frame(right\_frame,bd=2,relief=RIDGE)

table\_frame.place(x=5,y=70,width=700,height=300)

# saveas\_btn=Button(btn\_frame,text="save as",width=15,command=self.punctual,font=("times new roman",12,"bold"),bg="green")

# saveas\_btn.grid(row=1,column=2)

scroll\_x=ttk.Scrollbar(table\_frame,orient=HORIZONTAL)

scroll\_y=ttk.Scrollbar(table\_frame,orient=VERTICAL)

self.student\_table=ttk.Treeview(table\_frame,column=("Student\_id","Student\_Name","Course","Branch","Date","Time"),xscrollcommand=scroll\_x.set,yscrollcommand=scroll\_y.set)

scroll\_x.pack(side=BOTTOM,fill=X)

scroll\_y.pack(side=RIGHT,fill=Y)

scroll\_x.config(command=self.student\_table.xview)

scroll\_y.config(command=self.student\_table.xview)

self.student\_table.heading("Student\_id",text="Student\_id")

self.student\_table.heading("Student\_Name",text="Student\_Name")

self.student\_table.heading("Course",text="Course")

self.student\_table.heading("Branch",text="Branch")

self.student\_table.heading("Date",text="Date")

self.student\_table.heading("Time",text="TIme")

self.student\_table["show"]="headings"

self.student\_table.column("Student\_id",width=100)

self.student\_table.column("Student\_Name",width=100)

self.student\_table.column("Course",width=100)

self.student\_table.column("Branch",width=100)

self.student\_table.column("Date",width=100)

self.student\_table.column("Time",width=100)

self.student\_table.pack(fill=BOTH,expand=1)

def btech(self):

conn=mysql.connector.connect(host="localhost",user="root",password="",database="face\_recognizer")

my\_cursor=conn.cursor()

my\_cursor.execute("select DISTINCT \* from facedetails where course='btech'")

data=my\_cursor.fetchall()

if len(data)!=0:

self.student\_table.delete(\*self.student\_table.get\_children())

for i in data :

self.student\_table.insert("",END,values=i)

conn.commit()

conn.close()

def showall(self):

conn=mysql.connector.connect(host="localhost",user="root",password="",database="face\_recognizer")

my\_cursor=conn.cursor()

my\_cursor.execute("select DISTINCT \* from facedetails ")

data=my\_cursor.fetchall()

if len(data)!=0:

self.student\_table.delete(\*self.student\_table.get\_children())

for i in data :

self.student\_table.insert("",END,values=i)

conn.commit()

conn.close()

def random(self):

conn=mysql.connector.connect(host="localhost",user="root",password="",database="face\_recognizer")

my\_cursor=conn.cursor()

my\_cursor.execute("select \* from facedetails where time between '08:30:00' and '08:45:00' order by RAND() LIMIT 1 ")

data=my\_cursor.fetchall()

if len(data)!=0:

self.student\_table.delete(\*self.student\_table.get\_children())

for i in data :

self.student\_table.insert("",END,values=i)

conn.commit()

conn.close()

def defaulter(self):

conn=mysql.connector.connect(host="localhost",user="root",password="",database="face\_recognizer")

my\_cursor=conn.cursor()

my\_cursor.execute("select DISTINCT \* from facedetails where not time between '08:30:00' and '08:45:00'")

data=my\_cursor.fetchall()

if len(data)!=0:

self.student\_table.delete(\*self.student\_table.get\_children())

for i in data :

self.student\_table.insert("",END,values=i)

conn.commit()

conn.close()

def csv(self):

import csv

import mysql.connector

conn=mysql.connector.connect(host="localhost",user="root",password="",database="face\_recognizer")

cur = conn.cursor()

#file = open('students.csv')

file = open('D:/face recognition - Copy/attendance/mark.csv')

csv\_data = csv.reader(file)

skipHeader = True

for row in csv\_data:

if skipHeader:

skipHeader = False

continue

cur.execute('INSERT INTO facedetails(Student\_id, Student\_name,Course,Branch,Date,Time)' 'VALUES(%s,%s, %s, %s, %s, %s)', row)

query = "LOAD DATA INFILE 'D:/face recognition - Copy/attendance/mark.csv' INTO TABLE attendance FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' (Student\_id, Student\_name,Course,Branch,Date,Time)"

cur.execute(query)

messagebox.showinfo("sucess","exported csv succesfully",parent=self.root)

conn.commit()

conn.close()

def punctual(self): conn=mysql.connector.connect(host="localhost",user="root",password="",database="face\_recognizer")

my\_cursor=conn.cursor()

my\_cursor.execute("select DISTINCT \* from facedetails where time between '08:30:00' and '08:45:00'")

data=my\_cursor.fetchall()

if len(data)!=0:

self.student\_table.delete(\*self.student\_table.get\_children())

for i in data :

self.student\_table.insert("",END,values=i)

conn.commit()

conn.close()

def mca(self): conn=mysql.connector.connect(host="localhost",user="root",password="",database="face\_recognizer")

my\_cursor=conn.cursor()

my\_cursor.execute("select DISTINCT \* from facedetails where course='mca'")

data=my\_cursor.fetchall()

if len(data)!=0:

self.student\_table.delete(\*self.student\_table.get\_children())

for i in data :

self.student\_table.insert("",END,values=i)

conn.commit()

conn.close()

def mtech(self): conn=mysql.connector.connect(host="localhost",user="root",password="",database="face\_recognizer")

my\_cursor=conn.cursor()

my\_cursor.execute("select \* from facedetails where course='mtech'")

data=my\_cursor.fetchall()

if len(data)!=0:

self.student\_table.delete(\*self.student\_table.get\_children())

for i in data :

self.student\_table.insert("",END,values=i)

conn.commit()

conn.close()

def mba(self): conn=mysql.connector.connect(host="localhost",user="root",password="",database="face\_recognizer")

my\_cursor=conn.cursor()

my\_cursor.execute("select DISTINCT \* from facedetails where course='mba'")

data=my\_cursor.fetchall()

if len(data)!=0:

self.student\_table.delete(\*self.student\_table.get\_children())

for i in data :

self.student\_table.insert("",END,values=i)

conn.commit()

conn.close()

if \_\_name\_\_=="\_\_main\_\_":

root=Tk()

obj=Report(root)

root.mainloop()

**CHAPTER - 6**

**SYSTEM TESTING**

System Testing is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system. Ultimately, the software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

System Testing involves testing the software code for following.

* Testing the fully integrated applications including external peripherals in order to check how components interact with one another and with the system as a whole. This is also called the End to End testing scenario.
* Verify thorough testing of every input in the application to check for desired outputs.
* testing of the user’s experience with the application.

**6.1 Preparation of Test Data & Test Oracle:-**

Test Data in Software Testing is the input given to a software program during test execution. It represents data that affects or affected by software execution while testing. Test data is used for both positive testing to verify that functions produce expected results for given inputs and for negative testing to test software ability to handle unusual, exceptional or unexpected inputs.

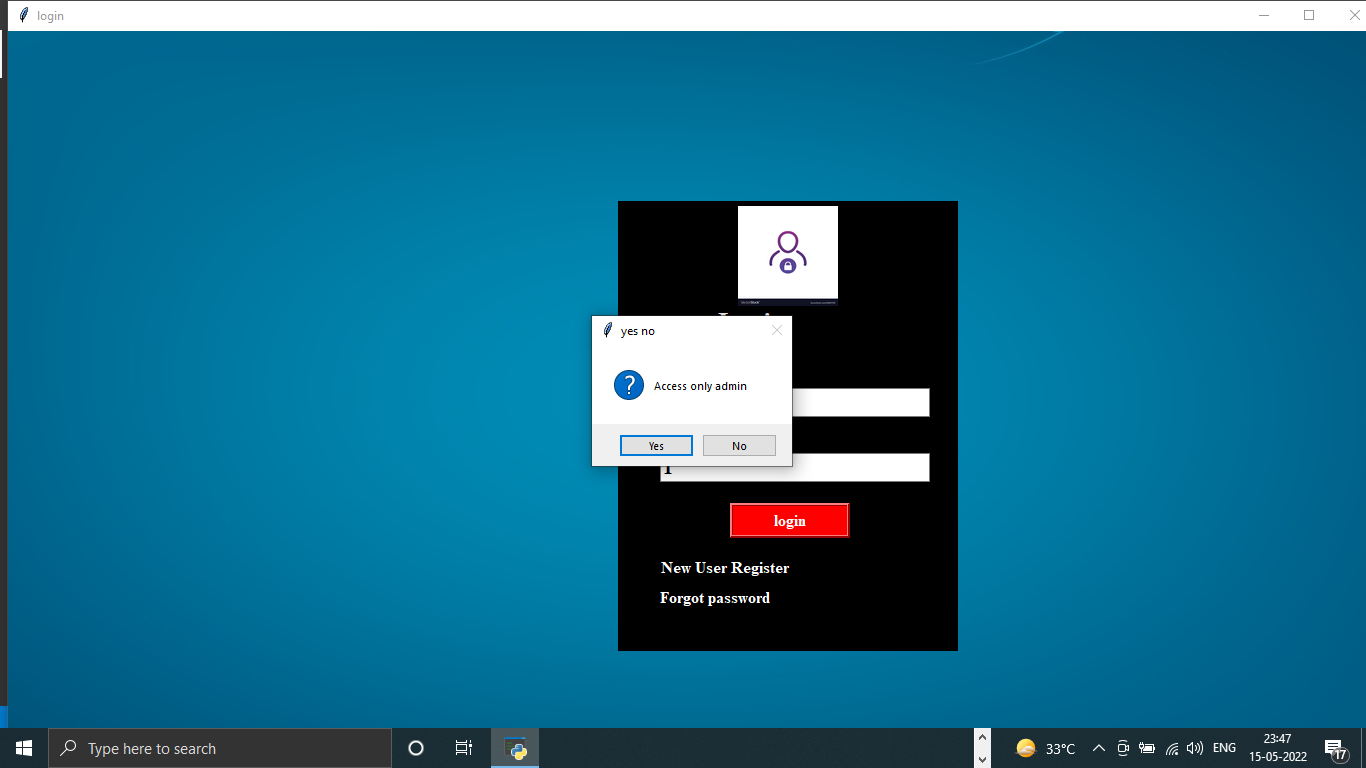
Poorly designed testing data may not test all possible test scenarios which will hamper the quality of the software.

**6.2 Unit Testing, Class Testing :-**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

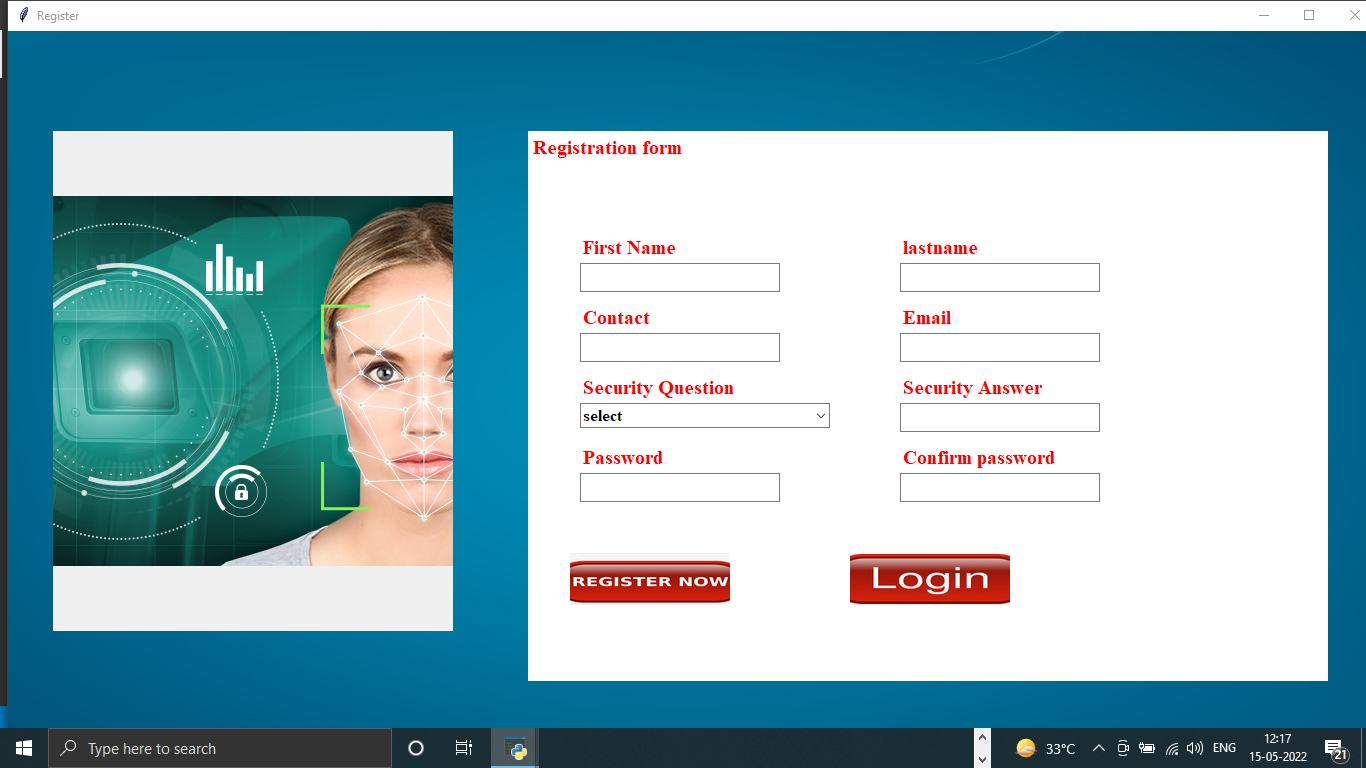
**Class Testing** :- Class testing is the base of object-oriented software testing. It involves three aspects: testing each method, testing the relations among class methods and testing the inheriting relation between class and subclass.

**6.2 Test with live data**

Firstly admin enter the email and password then if authentication of data is correct then enter the main page of face recognition modules

If admin want to add new staff for handling the face recognition software modules

Then firstly register the new admin staff



Then register successfully then admin can add student details like (Student id ,college id,Course,Semester,Branch,Mobile no ,Date of birth etc )

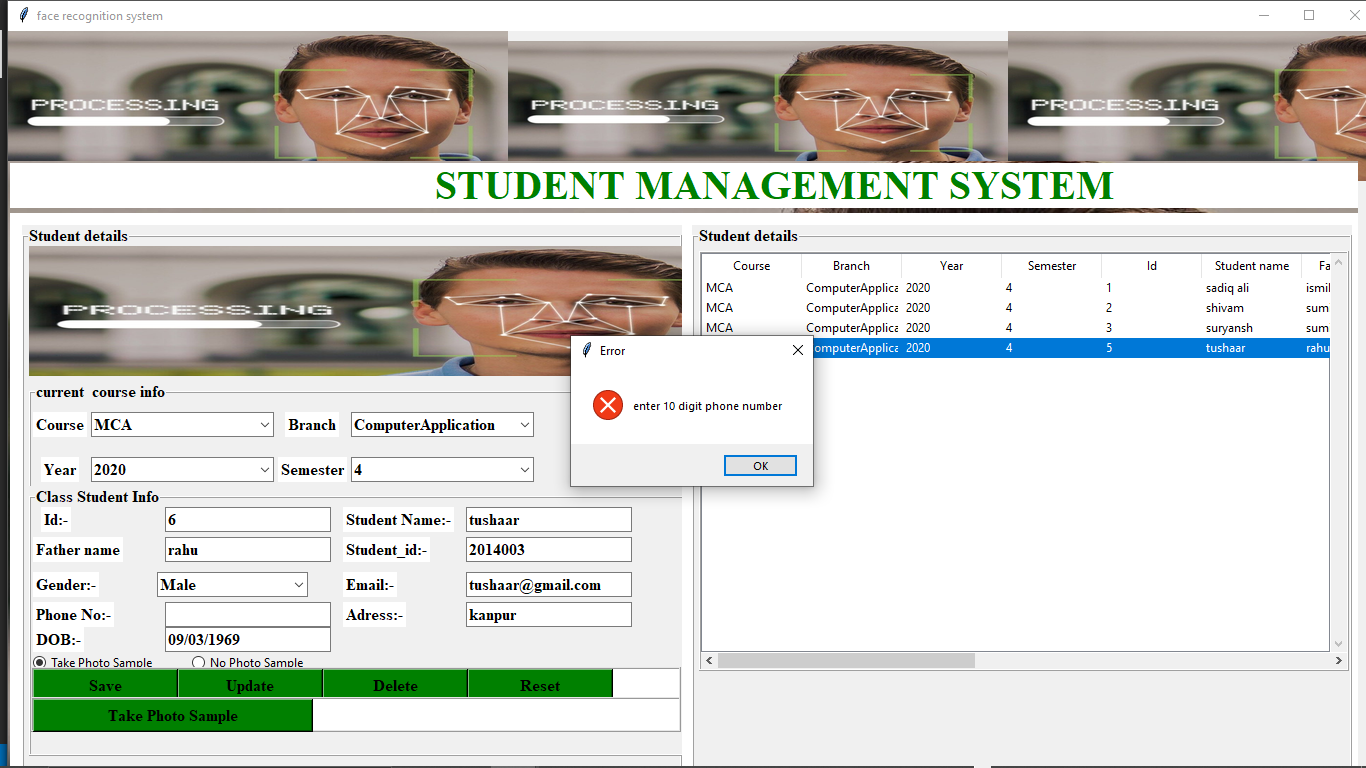
And also take the photo sample of student

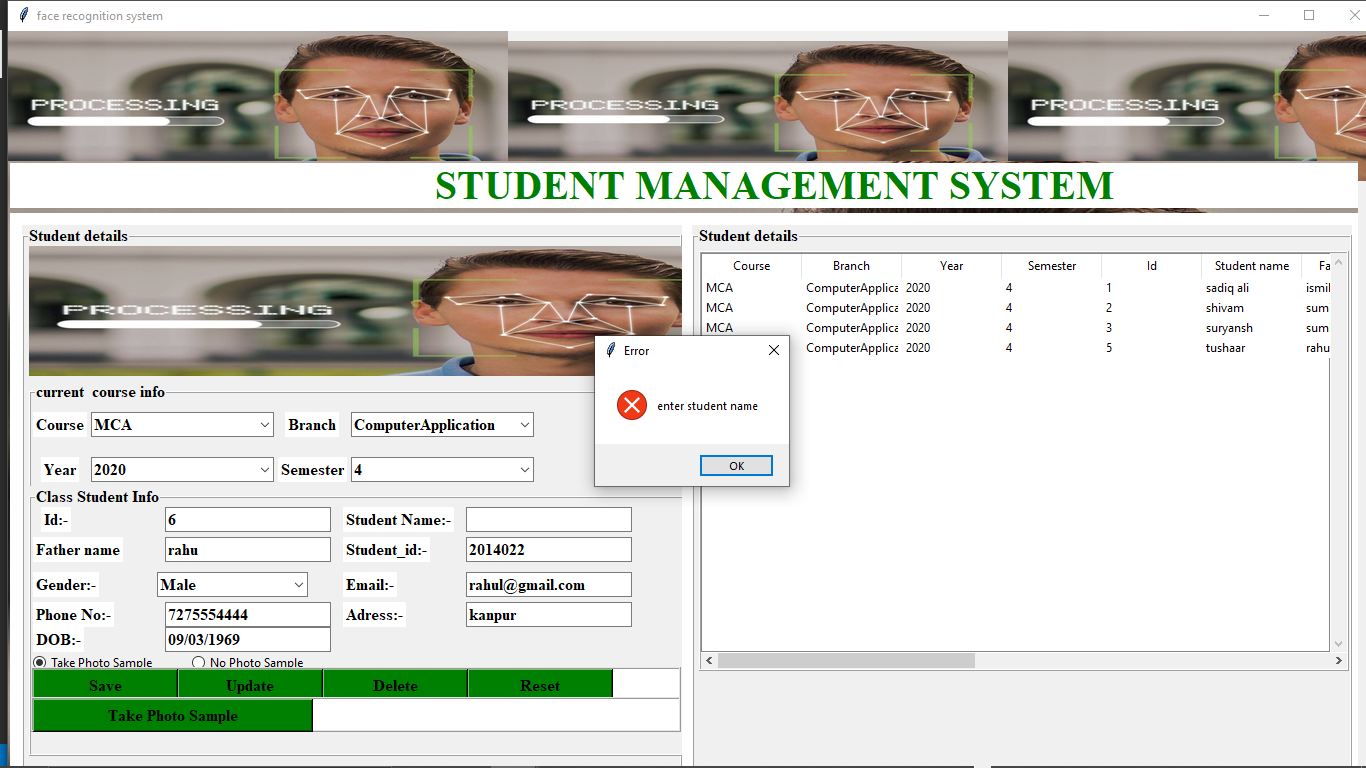
If admin want to take the photo then firstly click on the right side of table then choose the student details then click the student to want to add photo sample

If it selected the wrong the student details then all details of student are mixed up if not proper selected the right students details

First of all field are required to fill any new registration of student

All field are using validation then any one are none or empty then generate error then proper show the what type of error in student details form





**CHAPTER - 7**

**SYSTEM SECURITY**

**7.1 Checks and Control**

Encryption

Image result for encryption and secure in face recognition

with biometric encryption, rather than storing one's facial im- age in a database, the facial image is used to encrypt (code) some other information such as a cryptographic key and only. the biometrically-encrypted data is stored.

Secure

Unlike other identification solutions such as passwords, verification by email, selfies or images, or face identification, facial recognition uses unique mathematical and dynamic patterns that make this system one of the safest and most effective ones.

**CHAPTER - 8**

**Limitation And Special Feature**

**Limitation :**

While there are many advantages to the Punctuality Drive Using Face Recognition , there are also some disadvantages to Punctuality Drive Using Face Recognition. They are:

1. Poor Image Quality Limits Facial Recognition's Effectiveness

Image quality affects how well facial-recognition algorithms work. The image quality of scanning video is quite low compared with that of a digital camera. Even high-definition video is, at best, 1080p (progressive scan); usually, it is 720p. These values are equivalent to about 2MP and 0.9MP, respectively, while an inexpensive digital camera attains 15MP. The difference is quite noticeable.

2. Small Image Sizes Make Facial Recognition More Difficult

When a face-detection algorithm finds a face in an image or in a still from a video capture, the relative size of that face compared with the enrolled image size affects how well the face will be recognized. An already small image size, coupled with a target distant from the camera, means that the detected face is only 100 to 200 pixels on a side. Further, having to scan an image for varying face sizes is a processor-intensive activity. Most algorithms allow specification of a face-size range to help eliminate false positives on detection and speed up image processing.

3. Different Face Angles Can Throw Off Facial Recognition's Reliability

The relative angle of the target’s face influences the recognition score profoundly. When a face is enrolled in the recognition software, usually multiple angles are used (profile, frontal and 45-degree are common). Anything less than a frontal view affects the algorithm’s capability to generate a template for the face. The more direct the image (both enrolled and probe image) and the higher its resolution, the higher the score of any resulting matches.

4. Data Processing and Storage Can Limit Facial Recognition Tech

Even though high-definition video is quite low in resolution when compared with digital camera images, it still occupies significant amounts of disk space. Processing every frame of video is an enormous undertaking, so usually only a fraction (10 percent to 25 percent) is actually run through a recognition system. To minimize total processing time, agencies can use clusters of computers. However, adding computers involves considerable data transfer over a network, which can be bound by input-output restrictions, further limiting processing speed.

### Special Features

**AN AIRTIGHT MATCHING ALGORITHM**

A lot of facial matching algorithms just use tens or hundreds of feature detection points on a face in order to establish identity. But an enterprise face recognition solution should be using thousands of points on a face in order to establish identity. A better algorithm can result in a much higher degree of accuracy, which will undoubtedly result in a better ROI.

**SCALABILITY**

Enterprise facial recognition solutions have to be able to quickly scale flawlessly across hundreds (or even thousands) of locations. This means that you need to find a solution that is built to handle large deployments. The solution should also have a support team in place to handle installations, including optimizing cameras for lighting conditions and angle.

**BUILT-IN PRIVACY PROTECTION**

An enterprise face recognition company shouldn’t just think about its customers, but also its customers’ customers. That’s why it’s important to invest in a face detection system that is built to protect privacy. Some privacy-related features to look for include:

Data encryption– Image data is encrypted both at rest and during transmission.

Data breach precautions– Biometric templates stored within the facial recognition system should never be converted back into a face image in the case of a data breach.

Data purging– Surveillance data should be automatically purged at regular intervals in accordance with industry best practices

Anti-profiling– Facial recognition systems should be built to prevent profiling by race, age, gender or national origin.

As with any technology, there are potential drawbacks to using facial recognition, such as threats to privacy, violations of rights and personal freedoms, potential data theft and other crimes. There's also the risk of errors due to flaws in the technology.

**CHAPTER - 9**

**CONCLUSION**

The proposed method uses face detection and face recognition that helps to maintain the automated face recognition system system. For detection, (HAAR) is used

and for face recognition Linear Binary Pattern Histogram (LPBH) algorithm is applied.

In the result, the unique ID and name of the student is displayed along with the confidence percentage. Confidence percentage represents the distance between the histogram of the stored image and histogram of the real time image and is calculated by using Euclidean distance. Lower is the distance, higher is the recognition rate.

This system aims to build an effective class attendance

system using face recognition techniques. The proposed

system will be able to mark the attendance via face Id. It will

detect faces via webcam and then recognize the faces. After

recognition, it will mark the attendance of the recognized

student and update the attendance record

**CHAPTER -10**

**FUTURE SCOPE**

Preventing the frauds at ATMs in India. A database of all customers with ATM cards in India can be created and facial recognition systems can be installed. So, whenever user will enter in ATM his photograph will be taken to permit the access after it is being matched with stored photo from the database.

Reporting duplicate voters in India.

Passport and visa verification can also be done using this technology.

Also, driving license verification can be done using the same approach.

In defence ministry, airports, and all other important places the technology can be used to ensure better surveillance and security.

It can also be used during examinations such as Civil Services Exam, SSC, IIT, MBBS, and others to identify the candidates.

This system can be deployed for verification and attendance tracking at various government offices and corporates.

For access control verification and identification of authentic users it can also be installed in bank lockers and vaults.

For identification of criminals the system can be used by police force also.

**CHAPTER 11**

**BIBLOGRAPHY**

[1]Chaitra T.K, M.C. Chandrashekhar, and Dr. M.Z. Kurian

Attendance Management System Using Face Recognition,

ISSN: 0976-2876 (Print) ISSN: 2250-0138(Online).

[2] Kaggle, Google, Stack Overflow, Youtube.

[3] S. Agarwal and D. Roth. Learning a Sparse Representation for Object Detection. In European Conference on Computer Vision, 2004.

[4] Sneha D. Ikhar, Sagar M. Bhakre and Vaishali M. Bodhe

Student Attendance System by Face Detection

ISO 3297:2007 Certified Vol. 6, Issue 1, January 2017

[5] V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios. Boostmap: A method for efficient approximate similarity rankings. In IEEE Conf. on Computer Vision and Pattern Recognition, Madison, WI, June 2004